ARM® CoreLink™ MMU-400 System Memory Management Unit

Revision: r0p1

Technical Reference Manual

ARM®

ARM DDI 0472B (ID032014)

ARM CoreLink MMU-400 System Memory Management Unit
Technical Reference Manual

Release Information

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

Change history

Date	Issue	Confidentiality	Change
07 October 2011	A	Non-Confidential	First release for r0p0
20 March 2014	B	Non-Confidential	First release for r0p1

Proprietary Notice

Confidentiality Status

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

Product Status

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

Web Address

`http://www.arm.com`

Contents
ARM CoreLink MMU-400 System Memory Management Unit Technical Reference Manual

Preface

This preface introduces the *ARM®CoreLink™ MMU-400 System Memory Management Unit Technical Reference Manual*. It contains the following sections:

About this book

This book is for the MMU-400.

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 device that uses the MMU-400.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the MMU-400 and its features.

Chapter 2 *Functional Description*

Read this for an overview of the major functional blocks and the operation of the MMU-400.

Chapter 3 *Programmers Model*

Read this for a description of the MMU-400 memory map and registers.

Appendix A *Signal Descriptions*

Read this for a description of the MMU-400 signals.

Appendix B *Revisions*

Read this for a description of the technical changes between released issues of this book.

Glossary

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

See the *ARM® Glossary*,
http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html.

Conventions

Conventions that this book can use are described in:

- *Typographical conventions* on page vii.
- *Signals* on page vii.

Typographical conventions

The following table describes the typographical conventions:

Typographical conventions

Style	Purpose
italic	Introduces special terminology, denotes 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.
`monospace`	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>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

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.

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

Additional reading

This section lists publications by ARM and by third parties.

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

ARM publications

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

- *ARM® System Memory Management Unit Architecture Specification* (ARM IHI 0062).
- *ARM® CoreSight™ Architecture Specification* (ARM IHI 0029).
- *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* (ARM DDI 0406).
- *ARM® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite* (ARM IHI 0022).

The following confidential books are only available to licensees:

- *ARM ®CoreLink™ MMU-400 System Memory Management Unit AMBA® Designer (ADR-400) User Guide Supplement* (ARM DSU 0017).
- *ARM® CoreLink™ MMU-500 System Memory Management Unit Technical Reference Manual Supplement* (ARM DSU 0030).
- *ARM ®CoreLink™ MMU-400 System Memory Management Unit Implementation Guide* (ARM DII 0265).
- *ARM ®CoreLink™ MMU-400 System Memory Management Unit Integration Manual* (ARM DII 0266).

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 0472B.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Note

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

Chapter 1
Introduction

This chapter provides an overview of the MMU-400. It contains the following sections:

1.1 About the MMU-400

The *ARM® CoreLink™ MMU-400* provides ARM v7 virtualization extensions to bus masters in the system other than the processor. The MMU-400 translates addresses in hardware to accelerate hypervisor software virtualization of multiple guest *Operating Systems* (OS)s. You can also configure the MMU-400 to optimize the features, performance, and gate count required for the intended applications.

An address translation from an input address to an output address is described as a stage of address translation.

The MMU-400 supports the translation table formats defined by the ARM architecture, ARMv7, and can perform stage 2 translations that translate an input *intermediate physical address* (IPA) to an output *physical address* (PA). The MMU-400 uses inputs from the requesting master to identify a context. This context communicates to the MMU-400 what resources to use for the translation including which translation tables to use.

Figure 1-1 shows a single MMU-400 in an example system, performing address translation functions for a *Direct Memory Access* (DMA) controller.

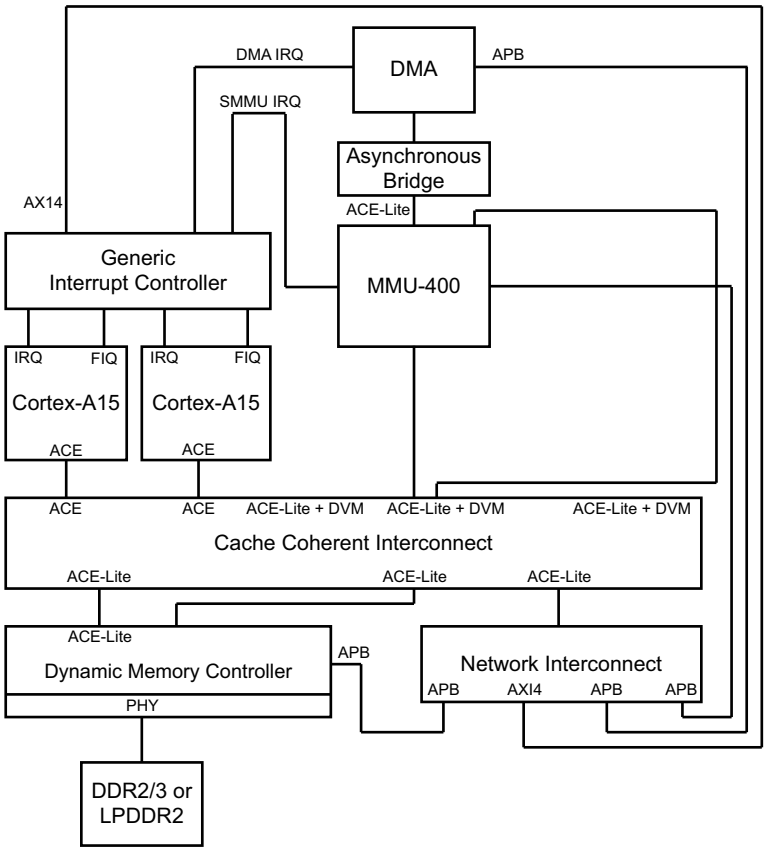


Figure 1-1 MMU-400 in system context

The MMU-400 checks access permissions, translates addresses, and provides the capability to generate or merge access attributes. The *Translation Look-aside Buffer* (TLB) maintenance is performed through:

- *Distributed Virtual-memory Messaging* (DVM) signaling.
- Programmable control registers.

The following are example masters for the MMU-400:

- *Graphics Processor Unit* (GPU).
- Video engines.
- DMA controllers.
- *Color LCD* (CLCD) controllers.
- Network controllers.

The MMU-400 routes each translation through the following logical processing steps:

- Security state determination.
- Context determination.
- Page table walk, if the translation is not cached in the TLB.
- Protection checks.
- Attribute generation or merging, depending on the programming.

You can configure the MMU-400 to bypass the translation process for a transaction or to fault a transaction regardless of the translation state.

1.2 Features of the MMU-400

The MMU-400 provides the following features:

- Converts from 32-bit to 40-bit *Large Physical Address Extension* (LPAE) addresses for 32-bit IO devices.
  - For more information on LPAE addresses, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.
- Multiple transaction contexts to apply to address translations for specific streams of transactions.
  - Supports up to eight configurable contexts. The MMU-400 maps each context by using an input StreamID from the master device that requires address translation.
- Stage 2 translations that translate an input IPA to an output PA in the ARM LPAE format.
  - For more information on LPAE addresses, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.
- Translation support for the ARMv7 4KB, 2MB, and 1GB page sizes:
  - For more information on ARM v7 virtualization extensions, LPAE addresses, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.
- Page table walk cache for storing intermediate page table walk data.
- Page table entry cache in the TLB.
- Support for TLB *Hit-Under-Miss* (HUM).
  - Up to four parallel page table walks.
- TLB invalidation through the AMBA 4 DVM signaling or register programming.
  - For more information on the DVM signaling, see the *ARM® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite*.
- Translation and protection check support including TrustZone® extension support.
- Fault handling, logging, and signaling that excludes demand paging support.
- Debug and performance-monitoring events.
- One AMBA slave interface that supports any of the following per *translation buffer unit* (TBU) for connecting the bus master device that requires address translations:
  - AXI3 protocol.
  - AXI4 protocol.
  - ACE-Lite.
- One AMBA master interface for master device transactions or *Page Table Walks* (PTWs) that support any of the following:
  - AXI3 protocol.
  - AXI4 protocol.
  - ACE-Lite with optional DVM extensions that supports the AXI data width, which are 64 or 128 bits with a configurable depth write buffer.
- An APB interface for programming that supports either of the following:
  - Separate APB3 interfaces for Secure and Non-secure programming.
  - A single APB4 interface for both Secure and Non-secure programming>.

The MMU-400 is based on the *ARM® System MMU Architecture Specification*.

1.2.1 Performance Targets

The typical TLB hit access latency is 2-3 cycles and the miss latency depends on memory sub-system infrastructure. The following table lists the process technology frequency targets.

Table 1-1 Process technology frequency targets

Process Technology	Frequency Targets	
	Min	Max
CP32LP	400Mhz	533Mhz

1.3 Configurable options

The following table lists the design-time features and configuration options supported by the MMU-400.

Table 1-2 Design-time features and configuration options

Feature	Min	Max
Stage 1 translation (VA→IPA)[a]	No	No[b]
Stage 2 translation (IPA→PA)[a]	Yes	Yes
Stage 1 and Stage 2 (VA→PA)[a]	No	No[b]
Translation contexts	1	8
Profiling resource banks	3	3
TLB entries	2	64
ACE-TLB invalidation through DVM[a]	No	Yes
Parallel PTWs	4	4
Translation prediction/pre-fetch[a]	Yes	Yes
Debug TLB visibility[a]	Yes	Yes
Software TLB loading[a]	No[c]	No[c]
Page table walk (QoS aware)[a]	Yes	Yes
Outstanding transactions	1	128
Input/Output port bus width	64	128
Input ports ACE-Lite + DVM[d]	0	1
Output ports ACE-Lite	0	1
Programming port (APB4 or APB3)	1	1
Generation of error responses[a]	Yes	Yes

a. Where:
   - **Yes** Indicates that the MMU-400 supports the feature.
   - **No** Indicates that the MMU-400 does not support the feature.

b. The MMU-500 supports stage 1 and stage 1 followed by stage 2 translations.

c. The direct TLB control is not recommended for security reasons.

d. The ACE-Lite + DVM is a subset of AMBA 4 ACE, the AMBA Coherency Extensions. You can tie-off the unused signals.

1.4 Product revisions

This section describes the differences in functionality between product revisions of the MMU-400:

r0p0 First release.

r0p0 - r0p1 No functional changes.

Chapter 2
Functional Description

This section describes the functional operation of the MMU-400. It contains the following sections:

2.1 About the functions

The TLB and PTW are the major functional blocks of the MMU-400. The TLB caches frequently used address ranges and the PTW performs page table walks.

Figure 2-1 shows the block diagram for the MMU-400.

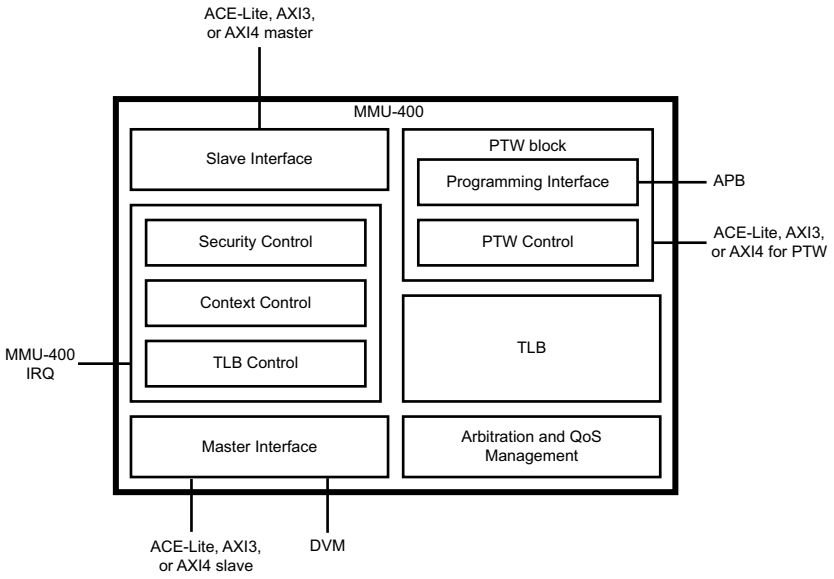


Figure 2-1 MMU-400 block diagram

The MMU-400 applies the following logical processing steps to every transaction:

1. Determines the security state of the device that originates the transaction. The security attribute presented on **AWPROT[1]** and **ARPROT[1]** signals is different from the security state of the device. Identifying the security state of the device is called security state determination.
2. Maps an incoming transaction to one of the contexts using an incoming StreamID.
3. Caches frequently used address ranges using the TLB. The best-case hit latency of this caching is two clocks when the TBU address slave register slices are not implemented. The best-case latency is three clocks when the TBU address slave register slices are specified.
4. Performs the main memory PTW automatically on a TLB address miss.
   - The MMU-400 shares the page table formats with the processor as specified in the LPAE for maximum efficiency.

     For more information on LPAE addresses, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.
5. Applies the required fault handling for every transaction.
6. Performs debug and performance monitoring through programmable performance counters that report statistics, for example, TLB refills or number of read or write accesses.

2.2 Interfaces

The MMU-400 contains the following interfaces:

- *AXI interfaces*.
- *Programming interface*.
- *ACE-Lite interfaces* on page 2-4.
- *Low-power interface for clock gating* on page 2-5.

2.2.1 AXI interfaces

The MMU-400 supports the following AXI interfaces:

- *AXI master interface*.
- *AXI slave interface*.

AXI master interface

The AXI master interface, with _m suffix, drives the translated address to the downstream slave. You must connect pin-to-pin the read address, write address, read data, write data, and buffered write response channels to the corresponding AXI slave interface.

If the MMU-400 is configured to support a dedicated interface for PTWs, you must connect the read address and read data channels of the slave interface associated with the PTWs to the MMU-400 PTW channel. In this configuration, the PTW channel contains the _ptw suffix. For example, **araddr_ptw** and **acaddr_ptw**.

AXI slave interface

The AXI slave interface, with _s suffix, drives the untranslated address to the TBU. You must connect pin-to-pin the read address, write address, read data, write data, and buffered write response channels of the AXI slave interface to an AXI master interface. In a system, the master interface can be the AXI bus infrastructure output or the output of a bridge that converts another bus protocol to AXI.

There must be AXI type compatibility between the MMU-400 and the master connected to the MMU-400.

2.2.2 Programming interface

The MMU-400 uses the APB interface as a programming interface, to permit the software to program the registers and to perform the debug operation.

The MMU-400 provides one of the following interfaces, which is selected during the MMU-400 configuration:

- The APB4 programming interface. For more information on MMU-400 integration into the AMBA 4 system, see the *ARM® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite*.
- Two APB3 programming interfaces. One of the APB interfaces is configured as Secure, and other interface is configured as Non-secure. You must ensure that only:
  - The Secure transactions are sent on the Secure interface.
  - The Non-secure transactions are sent on the Non-secure interface.

  For more information, see the *ARM® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite*.

APB slave interface

You must connect the APB slave interface to an AMBA 3 APB or AMBA 4 APB master, depending on the APB configuration type.

If you have configured the AMBA 3 APB interface, there are two ports with _s suffix for Secure register accesses and an _ns suffix for Non-secure register accesses.

The MMU-400 provides a 32-bit address bus, **paddr[31:0**], but it only uses bits[15:2]. The MMU-400 ignores:

- Bits[31:16], but their presence facilitates the MMU-400 integration with adjacent RTL blocks, such as an interconnect.
- Bits[1:0], because the smallest accesses allowed to its internal registers are word accesses.

If the APB is not running at the same frequency as that of the **cclk** signal, then you can divide the frequency using the **pclken** signal value.

When operating at this frequency, the APB master must also use the **pclken** signal so that the signal states change only on enabled clock edges. If the APB interface is to run at the same frequency as that of the **cclk** signal, you must ensure that the **pclken** signal is tied HIGH.

2.2.3 ACE-Lite interfaces

You can configure the MMU-400 to use one of the AXI3, AXI4, or ACE-Lite interfaces to receive and forward transactions, and to perform page table walks. The interface information is described in the following sections:

- *AXI3 interface*.
- *AXI4 interface* on page 2-5.
- *ACE-Lite interface* on page 2-5.
  - *TLB block barrier support* on page 2-5.

AXI3 interface

The MMU-400 can be configured to support the AXI3 protocol. In this mode, only AXI3 signaling is present on the main data path through the MMU-400.

The MMU-400 supports DVM messages only if a dedicated ACE-Lite master port is configured to perform PTWs.

The following features of AXI3 are not supported in the MMU-400:

- Write data interleaving.

  Note

  Write data and write address ordering must be the same, otherwise data corruption can occur.

- Locked transfer. The **AxLOCK[1**] signal bit cannot be set to 1. However, if any transaction has its **AxLOCK[1]** set, then it is ignored, and the output transaction has the **AxLOCK[1]** reset. The other bits of the **AxLOCK** signal are supported.

  Note

  If the MMU-400 receives a locked transaction, the output transaction is passed to the downstream slave as a normal access.

AXI4 interface

The MMU-400 can be configured to support the AXI4 protocol. In this mode, only the AXI4 signaling is present on the main data path through the MMU-400.

The MMU-400 supports DVM messages only if a dedicated ACE-Lite master port is configured to perform PTWs.

ACE-Lite interface

The MMU-400 can be configured to support the ACE-Lite protocol. In this mode, only the ACE-Lite signaling is present on the main data path through the MMU-400.

The MMU-400 supports DVM messages with a combined master port, or when a dedicated ACE-Lite master port is configured to perform PTWs.

When you configure the MMU-400 to support the ACE-Lite interface, you must connect the AC channel to the CCI-driven AC channel or to the ACE-compatible slave interface that supports DVM messaging.

ARM recommends that you use the DVM channel for TLB maintenance operations. If the system cannot access the DVM channel, you must tie the **ACVALID** signal LOW and use the programming interface for TLB maintenance operations.

When you configure the MMU-400 to provide a dedicated AXI channel to perform PTWs, the ACVALID channel must be part of the PTW.

TLB block barrier support

A TLB block in the MMU-400 receives, passes on, and generates barriers of its own, in response to the **SYNC** signal received from the DVM channel of the PTW block.

The PTW block sends the **SYNC** signal to the TLB block on receiving one of the following:

- The SYNC message received on the programming interface.
- The DVM **SYNC** message.

For more information on SYNC and DVM SYNC messages, see the *ARM® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite*.

2.2.4 Low-power interface for clock gating

The MMU-400 has two AXI low-power interfaces each of which can be connected pin-to-pin to a dedicated interface of a clock controller. You can use the low-power interfaces to disable the clock of each sub-block of the MMU-400. Alternatively, if there is no clock controller, you must tie the **csysreq_*** inputs HIGH, and can leave the outputs, **csysack_*** and **cactive_*** unconnected.

For more information on the signal functionality, see the *ARM® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite* .

Note

You must gate the clock when the **csysreq_*** and **csysack_*** signals are LOW.

The MMU-400 has two AXI LPIs that allow you to disable the clock for the PTW and TLB blocks independently, as the following sections describe:

- *TLB block* on page 2-6.
- *PTW block* on page 2-6.

TLB block

An external clock controller can request the TLB block to enter the low-power state by de-asserting the **csysreq_tbu** signal.

The TLB block can enter the low-power state under the following conditions:

- No outstanding access is pending.
- No input access is pending.
- No TLB maintenance operation is pending.

When the preceding conditions are satisfied, the **cactive_tbu** signal is set LOW. The low-power entry request is acknowledged when the **csysreq_tbu** signal goes LOW. You must disable the clock when the **csysack_tbu** and **cactive_tbu** signals are LOW. For more information on the AXI LPI signals, see *AXI low-power interface signals* on page A-17.

PTW block

During normal operation, the PTW block is in the idle state. Therefore ARM recommends that you use the AXI low-power interface to disable the clock for this block.

An external clock controller can request the PTW block to enter into the low-power mode by de-asserting the **csysreq_tcu** signal.

The PTW block pulls the **cactive_tcu** signal LOW when all the following conditions are satisfied:

- The **PSELx** signal is LOW.
- The **ACVALID** signal is LOW.
- No outstanding PTW request is pending.

The **PSELx** signal is combinatorially connected to the **cactive_tcu** signal. Therefore, if the APB interface is synthesized at a value less than that of the **cclk** signal using the **pclken** signal, treat the **cactive_tcu** signal as asynchronous to prevent metastability problems in the clock controller.

The PTW block acknowledges the low-power entry request by setting the **csysack_tcu** signal LOW. You can disable the clock when the **csysack_tcu** and **cactive_tcu** signals are LOW. For more information on the AXI LPI signals, see *AXI low-power interface signals* on page A-17.

2.3 Generation of StreamID

A StreamID is used to map the incoming transaction to a context by using the stream mapping table. The characteristics of the StreamID are as follows:

- The width of the StreamID is selected during the MMU-400 configuration.
- The width of the StreamID is selected as a function of the incoming AXI ID that is used for read or write transactions and up to 8 sideband signals. Select the `StreamID - width of the sideband signal` parameter value from the range 1-15 bits.

The following signals are selected to generate the StreamID:

AWID/ARID Bits[22:0]. If the StreamID width is less than 23 bits, you can set only the valid bits to one.

R/W Bit[23]. If this bit is set, you can generate the StreamID with this bit set to one for writes and zero for reads.

Sideband signals You can have a maximum of eight sideband signals.

The bit positions of the StreamID stay the same for writes and reads, though different signals are used to arrive at the StreamID.

Example 2-1 Generation of StreamID - example 1

If bit[7], bit[4], bit[3], and bit[2] of **AWID**/**ARID**, the R/W bit, and four of the sideband signals are selected to generate the StreamID, the final generated StreamID has the following formats:

Writes `Stream_id = {wsb_sid[3:0],0b1,awid[7],awid[4],awid[3],awid[2]}`

Reads `Stream_id = {rsb_sid[3:0],0b0,arid[7],arid[4],arid[3],arid[2]}`

Where:

- **wsb_sid** and **rsb_sid** are the write and read sideband signals for the StreamID respectively.
- **awid** and **arid** are the write and read ID signals respectively.

Example 2-2 Generation of StreamID - example 2

If bit[2], bit[1], and bit[0] of **AWID**/**ARID** are selected to generate the StreamID, that is, there is no R/W bit and there are no sideband signals, the final generated StreamID has the following formats:

Writes `Stream_id = {awid[2],awid[1],awid[0]}`

Reads `Stream_id = {arid[2],arid[1],arid[0]}`

Where:

- **wsb_sid** and **rsb_sid** are the write and read sideband signals for the StreamID respectively.
- **awid** and **arid** are the write and read ID signals respectively.

2.4 Generation of the SSD index

The generation of the SSD index is similar to that of the StreamID.

The SSD index is selected during configuration. It is selected as a function of the incoming AXI ID, and up to eight sideband signals. Select the `SSD index signal width` parameter value from the range 1-15 bits.

The following signals are selected to generate the SSD index:

AWID/ARID Bits[22:0]. If the ID width is less than 23 bits, you can set the valid bits to one.

Sideband signals You can have a maximum of eight sideband signals.

The bit positions of SSD index stay the same for writes and reads, though different signals are used to arrive at the SSD index.

Example 2-3 Generation of StreamID - example 1

If bit[5], bit[3], bit[2], and bits[1:0] of **AWID**/**ARID** and two sideband signals are selected to generate the SSD Index, the final generated SSD index has the following formats:

Writes `SSD Index = {wsb_ssd[1:0], awid[5],awid[3],awid[2],awid[1],awid[0]}`

Reads `SSD Index = {rsb_ssd[1:0], arid[5],arid[3],arid[2],arid[1],arid[0]}`

Where:

- **wsb_ssd** and **rsb_ssd** are the write and read sideband signals for the SSD index respectively.
- **awid** and **arid** are the write and read ID signals respectively.

Example 2-4 Generation of StreamID - example 2

If bit[13], bit[10], and bit[5] of **AWID**/**ARID** are selected to generate the SSD index, that is, there are no sideband signals, the final generated SSD index has the following formats:

Writes `SSD Index = {awid[13],awid[10],awid[5]}`

Reads `SSD Index = {arid[13],arid[10],arid[5]}`

Where:

- **wsb_ssd** and **rsb_ssd** are the write and read sideband signals for the SSD index respectively.
- **awid** and **arid** are the write and read ID signals respectively.

2.5 Determining the security state of masters

When the SSD index is determined, the SSD table contains bits from 0 to $2^{(\text{SSD Index WIDTH})}-1$. You must determine the status of the bits as follows:

List of non-programmable indices

For these indices, the security state of the master is defined and does not change.

You must specify the indices of the masters whose security states are always Secure.

List of programmable indices

You can program the security state of the programmable indices.

You must determine the default state of each master whose security state is programmable.

Indices that are not included in the following states are considered as non-programmable non-secure indices:

- Non-programmable secure.
- Programmable.

Note

- An entry must not be duplicated in more than one list.
- You must specify at least one programmable or fixed Non-secure entry for every configuration.

Example 2-5 Programmable and non-programmable indices

If the SSD index width is 6 bits, there are 64 indices whose security states must be determined. For example:

Programmable secure default indices 1, 5, 9, 41, 60, 62

Programmable non-secure default indices 3, 7, 22, 42, 61, 63

Secure indices 24, 26, 28, 40

The indices between 0 and 63 that are not listed here are fixed as Non-secure indices.

2.6 *Hit-Under-Miss* (HUM)

HUM allows responses to the master if there is a TLB hit for a subsequent transaction while the MMU-400 is performing a translation for a previous transaction that had a TLB miss.

The HUM functionality is enabled by the write buffer.

HUM enables the hit transactions coming after miss transactions to be translated by the MMU-400. The hit transactions are translated only if the write data from the miss transactions can be accommodated in the write buffer.

The HUM has certain characteristics for read and write transactions:

- If the transactions are read access, HUM is automatically enabled.
- If the transactions are write operations, HUM is enabled or disabled based on the write buffer depth. You can specify the write buffer depth during configuration.
  - If the depth of the write buffer is zero, HUM is automatically disabled.
  - If the depth of the write buffer is non-zero, a write hit transaction is translated only if the write data from a missed transaction can be accommodated in the write buffer.

The number of outstanding missed transactions is determined by the depth of the write buffer. For example, if the depth of the buffer is four then it can hold two transactions of length two. Each buffer entry holds only one beat of the transaction, even if it is of a narrow width.

Example 2-6 shows a HUM condition.

Example 2-6 Hit under miss

Consider that the write buffer depth is eight and there are two missed write transactions of lengths four and three. Both missed write transactions are stored in the write buffer during the PTWs for the transactions. If you perform another transaction before the missed write transactions are processed, the new transaction is passed through, if that access results in a TLB hit.

Note

If the write buffer is full of missed transactions, HUM cannot occur.

2.7 Fault handling

The MMU-400 supports the terminate fault handling mode.

Note

The MMU-400 does not support the stall fault handling mode.

On a fault, the faulted transaction results in an abort based on the fault report setting in the following registers:

- The Secure Configuration Register for global faults.
- The System Control Register of that context for context faults.

When a fault occurs, the transactions that arrive after the faulted transaction can also fault if both of the following conditions are true:

- The second transaction is in the same 4K region and is in the same context as the first transaction.
- The second transaction is received before the response for the first transaction has been sent by MMU-400.

This fault can occur even if a fault clear is received between the first and second transactions.

Note

The MMU-400 does not log an external fault reported to the SMMU system in response to a transaction issued at a slave interface associated with any context bank. However, it logs a reported external fault that is synchronous to the SMMU system in response to a fetch issued as part of a translation table walk.

For information about the fault handling, see the *ARM® System Memory Management Unit Architecture Specification*.

2.8 Dynamic programming

ARM recommends that you modify the contents of any control register only when there are no outstanding transactions in the MMU-400. If any of the control registers are modified when there is an existing transaction in the MMU-400, then the following behaviors occur:

- When a control register is written, if a transaction arrives at the MMU-400 after the **PREADY**signal, the MMU-400 ensures that the new register attributes are applied to the transaction.
- When a control register is written, if a transaction is pending within MMU-400, it is unknown whether the old register attributes or new register attributes are applied to that transaction.

Chapter 3
Programmers Model

This chapter describes the MMU-400 registers and provides information about programming the MMU-400. It contains the following sections:

3.1 About the programmers model

The following information applies to the MMU-400 registers:

- Registers are implemented according to the *ARM® System Memory Management Unit Architecture Specification* with the security extensions implemented in the MMU-400 as follows:
  - *Global space 0 registers summary* on page 3-4.
  - *Global space 1 register summary* on page 3-6.
  - *Integration registers summary* on page 3-7.
  - *Performance monitoring registers summary* on page 3-7.
  - *The MMU-400 security state determination address space summary* on page 3-8.
  - *Peripheral and Component identification summary* on page 3-8.
  - *Translation context bank address map summary* on page 3-8.

The following information applies to the MMU-400 registers:

- Unless otherwise stated in the accompanying text:
  - Do not modify undefined register bits.
  - Ignore undefined register bits on reads.
  - All register values are UNKNOWN on reset unless otherwise stated.
- Access types in Table 3-1 on page 3-4 and Table 3-7 on page 3-8 are described as follows:

RW	Read and write.
RO	Read-only.
WO	Write-only.
RAZ	Read-As-Zero.
WI	Write-ignored.
RAO	Read-As-One.
RAZ/WI	Read-As-Zero, Writes Ignored.
RAO/SBOP	Read-As-One, Should-Be-One-or-Preserved on writes.
RAO/WI	Read-As-One, Writes Ignored.
RAZ/SBZP	Read-As-Zero, Should-Be-Zero-or-Preserved on writes.
SBO	Should-Be-One.
SBOP	Should-Be-One-or-Preserved.
SBZ	Should-Be-Zero.
SBZP	Should-Be-Zero-or-Preserved.

- When you configure registers using APB4, all transactions must be privileged and data. In other words, you must set **PPROT[0]**=1 and **PPROT[2]**=0, else these transactions are treated as RAZ/WI.

3.2 The MMU-400 address map

The MMU-400 is configured through a memory-mapped register frame. The total size of the MMU-400 address range depends on the number of implemented translation contexts.

The MMU-400 address map consists of the following equally sized portions:

The global address space

The global address space is located at the bottom of the MMU-400 address space, at SMMU_BASE. See Figure 3-1.

The translation context bank address space

The translation context bank address space is located above the top of the global address space, at SMMU_TOP. See Figure 3-1.

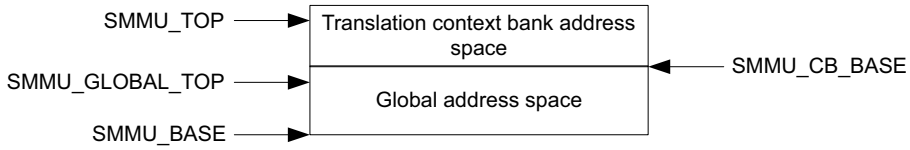


Figure 3-1 The MMU-400 address map, global space, and translation context bank

You can determine the size of the MMU-400 address range by reading the following register fields:

- SMMU_IDR1.PAGESIZE.
- SMMU_IDR1.NUMPAGENDXB.

For more information, see the *ARM® System Memory Management Unit Architecture Specification*.

3.3 Register summary

This section describes the registers of the MMU-400 in base offset order. The register map contains the following main blocks:

- *Global space 0 registers summary*.
- *Global space 1 registers summary* on page 3-6.
- *Integration register summary* on page 3-7.
- *Performance monitoring registers summary* on page 3-7.
- *The MMU-400 security state determination address space summary* on page 3-8.
- *Peripheral and Component identification registers summary* on page 3-8.
- *Translation context bank registers summary* on page 3-8.

Note

The following tables show the MMU-400 registers and provide a reference to the register description that either this book or the *ARM® System Memory Management Unit Architecture Specification* includes:

- *Global space 0 registers summary*.
- *Global space 1 registers summary* on page 3-6.
- *Integration register summary* on page 3-7.
- *Performance monitoring registers summary* on page 3-7.
- *The MMU-400 security state determination address space summary* on page 3-8.
- *Peripheral and Component identification registers summary* on page 3-8.
- *Translation context bank registers summary* on page 3-8.

3.3.1 Global space 0 registers summary

Table 3-1 shows the global space 0 registers in base offset order.

Note

The addresses that are not described in Table 3-1 are Reserved.

Table 3-1 Global space 0 registers summary

Name	Type	S or NS[a]	Offset	Description	Notes
SMMU_SCR0 SMMU_CR0	RW	S NS	0x00000	*Secure Configuration Register 0* on page 3-10	Banked with security
SMMU_SCR1	RW	S	0x00004	Secure Configuration Register 1	Secure only
SMMU_ACR	RW	NS	0x00010	*Auxiliary Configuration Register* on page 3-16	-
SMMU_SACR		S			Banked with security

Table 3-1 Global space 0 registers summary (continued)

Name	Type	S or NS[a]	Offset	Description	Notes
SMMU_IDR0	RO	S NS	0x00020	*Identification registers* on page 3-17	-
SMMU_IDR1		S NS	0x00024		-
SMMU_IDR2		S NS	0x00028		-
SMMU_IDR7		S NS	0x0003C		-
SMMU_SGFAR[31:0] SMMU_GFAR[31:0]	RW	S	0x00040	Global Fault Address Register	Banked with security
SMMU_SGFAR[63:32] SMMU_GFAR[63:32]			0x00044		
SMMU_GFSR	RW	NS	0x00048	Global Fault Status Register	-
SMMU_SGFSR		S			Banked with security
SMMU_GFSRRESTORE	WO	NS	0x0004C	Global Fault Status Register Restore	-
SMMU_SGFSRRESTORE		S			Banked with security
SMMU_GFSYNR0	RW	NS	0x00050	Global Fault Syndrome Register 0	-
SMMU_SGFSYNR0		S			Banked with security
SMMU_GFSYNR1	RW	NS	0x00054	Global Fault Syndrome Register 1	-
SMMU_SGFSYNR1		S			Banked with security
SMMU_STLBIALL	WO	S NS	0x00060	Invalidate entire TLB Register	Secure only
SMMU_TLBIVMID	WO	NS	0x00064	Invalidate TLB by VMID Register	-
SMMU_TLBIALLNSNH	WO	NS	0x00068	Invalidate Entire Non-secure Non-Hyp TLB Register	-
SMMU_STLBGSYNC	WO	S NS	0x00070	Global Synchronize TLB Invalidate Register	-
SMMU_TLBGSYNC					-
SMMU_STLBGSTATUS	RO	S NS	0x00074	Global TLB Status Register	Banked with security
SMMU_TLBGSTATUS					-
SMMU_DBGRPTR	RW	S	0x00080	*Debug registers* on page 3-23	-
SMMU_DBGRDATA	RO	S	0x00084		-

Table 3-1 Global space 0 registers summary (continued)

Name	Type	S or NS[a]	Offset	Description	Notes
SMMU_NSCR0	RW	S	0x00400	*Secure Alias to Non-secure Configuration Register 0* on page 3-27	Secure only
SMMU_NSACR	RW	S	0x00410	*Secure Alias to Non-secure Auxiliary Configuration Register* on page 3-27	Secure only
SMMU_NSGFAR[31:0][b]	RW	S	0x00440	Secure alias for Non-secure Global Fault Address Register	Secure only, 64-bit
SMMU_NSGFAR[63:32][b]	RW	S	0x00444	Secure alias for Non-secure Global Fault Address Register	
SMMU_NSGFSR[b]	WO	S	0x00448	Secure alias for Non-secure Global Fault Status Register	Secure only
SMMU_NSGFSRRESTORE[b]	WO	S	0x0044C	Secure alias for Non-secure Global Fault Status Register Restore	Secure only
SMMU_NSGFSYNR0[b]	RW	S	0x00450	Secure alias for Non-secure Global Fault Syndrome Register 0	Secure only
SMMU_NSGFSYNR1[b]	RW	S	0x00454	Secure alias for Non-secure Global Fault Syndrome Register 1	Secure only
SMMU_NSTLBGSYNC[b]	WO	S	0x00470	Secure alias for Non-secure Global Synchronize TLB Invalidate Register	Secure only
SMMU_NSTLBGSTATUS[b]	RO	S	0x00474	Secure alias for Non-secure Global TLB Status Register	Secure only
SMMU_SMR*n*	RW	S NS	0x00800-0x0087C	*Stream Match Registers* on page 3-27	-
SMMU_S2CR*n*	RW	S NS	0x00C00 - 0x00C7C	*Stream to Context registers* on page 3-29	-

a. S stands for Secure and NS stands for Non-secure.

b. Using Secure aliases, the Non-secure version of the banked registers are accessed.

3.3.2 Global space 1 registers summary

Table 3-2 shows the translation context bank address map in base offset order.

Table 3-2 Global space 1 register summary

Name	Type	S or NS	Offset	Description
SMMU_CBFRSYNRA0-7	RW	NS	0x1400-0x141C	*Context Bank Fault Restricted Syndrome Register A* on page 3-30
SMMU_CBAR0-7	RW	NS	0x1000-0x101C	*Context Bank Attribute Register* on page 3-31

3.3.3 Integration register summary

Table 3-3 shows the integration registers in base offset order.

Table 3-3 Integration registers summary

Name	Type	S or NS	Offset	Description
ITEN	RW	NS/S	0x2000	*Integration Enable Register* on page 3-33
ITIP	RO	NS/S	0x2004	*Integration Test Input Register* on page 3-34
ITOP	RW	NS/S	0x2008	*Integration Test Output Register* on page 3-34

3.3.4 Performance monitoring registers summary

Table 3-4 shows the performance monitoring registers in base offset order.

Table 3-4 Performance monitoring registers summary

Name	Type	S or NS	Offset	Description
PMEVCNTR0-2	RW	NS	0x3000-0x3008	*Performance Monitor Event Count Register* on page 3-36
PMEVTYPE0-2	RW	NS	0x3400-0x3408	Performance Monitor Event Type Select Register
PMCGCR*n*	RW	NS	0x3800	*Performance Monitor Counter Group Configuration Register* on page 3-37
PMCGSMR*n*	RW	NS	0x3A00	*Performance Monitor Counter Group Stream Match Register* on page 3-38
PMCNTENSET*n*	RW	NS	0x3C00	Performance Monitor Counter Enable Set and Clear registers
PMCNTENCLR*n*		NS	0x3C20	
PMINTENSET*n*	RW	NS	0x3C40	Performance Monitor Interrupt Enable Set and Clear registers
PMINTENCLR*n*			0x3C60	
PMOVSCLR*n*	RW	NS	0x3C80	Performance Monitor Overflow Status Set and Clear registers
PMOVSSET*n*			0x3CC0	
PMCFGR	RO	NS	0x3E00	*Performance Monitor Configuration Register* on page 3-39
PMCR	RW	NS	0x3E04	*Performance Monitor Control Register* on page 3-40
PMCEID0-1	RO	NS	0x3E20-0x3E24	Performance Monitor Common Event ID Register
PMAUTHSTATUS	RO	NS	0x3FB8	*Performance Monitor Authentication Status register* on page 3-41
PMDEVTYPE	RO	NS	0x3FCC	*Performance Monitor Device Type Register* on page 3-43

3.3.5 The MMU-400 security state determination address space summary

Table 3-5 shows the MMU-400 security state address space and its attributes.

Table 3-5 The MMU-400 security state determination address space summary

Name	Type	S or NS	Offset	Description
SMMU_SSDR0-1023	UNK/SBOP/WI/RO/RW[a]	S	0x04000-0x04FFC	*The MMU-400 Security State Determination Address Space* on page 3-44

a. If the SSD table is not implemented, the bits are UNK/SBOP.
If the SSD table is implemented, the non-defined bits are UNK/SBOP/WI and the defined bits are RO/RW as defined.

3.3.6 Peripheral and Component identification registers summary

Table 3-6 shows the Peripheral and Component identification registers in base offset order.

Table 3-6 Peripheral and Component identification summary

Name	Type	S or NS	Offset	Description
Periph ID 4	RO	NS/S	0x0FFD0	*Peripheral Identification registers* on page 3-46
Periph ID 5	RO	NS/S	0x0FFD4	
Periph ID 6	RO	NS/S	0x0FFD8	
Periph ID 7	RO	NS/S	0x0FFDC	
Periph ID 0	RO	NS/S	0x0FFE0	
Periph ID 1	RO	NS/S	0x0FFE4	
Periph ID 2	RO	NS/S	0x0FFE8	
Periph ID 3	RO	NS/S	0x0FFEC	
Component ID0	RO	NS/S	0x0FFF0	*Component Identification registers* on page 3-46
Component ID1	RO	NS/S	0x0FFF4	
Component ID2	RO	NS/S	0x0FFF8	
Component ID3	RO	NS/S	0x0FFFC	

3.3.7 Translation context bank registers summary

Table 3-7 shows the translation context bank address map in base offset order.

Table 3-7 Translation context bank address map summary

Name	Type	Size	Offset	Description
SMMU_CB*n*_SCTLR	RW	32	0x00000	*System Control Register* on page 3-50
SMMU_CB*n*_TTBR0[63:0]	RW	64	0x00020-0x00024	Translation Table Base Register
SMMU_CB*n*_TTBCR	RW	32	0x00030	*Translation Table Base Control Register* on page 3-53
SMMU_CB*n*_FSR[a]	-	-	0x00058	Fault Status Register. See the *ARM® System Memory Management Unit Architecture Specification*.

Table 3-7 Translation context bank address map summary (continued)

Name	Type	Size	Offset	Description
SMMU_CB*n*_FSRRESTORE[a]	-	-	0x0005C	Fault Status Restore Register. See the *ARM® System Memory Management Unit Architecture Specification*.
SMMU_CB*n*_FAR[31:0][a]	-	-	0x00060	Fault Address Register. See the *ARM® System Memory Management Unit Architecture Specification*.
SMMU_CB*n*_FAR[63:32][a]	-	-	0x00064	
SMMU_CB*n*_FSYNR0[a]	-	-	0x00068	Fault Syndrome Registers. See the *ARM® System Memory Management Unit Architecture Specification*.
SMMU_CB*n*_PMXEVCNTR*m*	-	-	0x00E00-0x00E08	See *Performance Monitoring registers* on page 3-36.
SMMU_CB*n*_PMXEVTYPER*m*	-	-	0x00E80-0x00E88	See *Performance Monitoring registers* on page 3-36.
SMMU_CB*n*_PMCFGR	-	-	0x00F00	See *Performance Monitoring registers* on page 3-36.
SMMU_CB*n*_PMCR	-	-	0x00F04	See *Performance Monitoring registers* on page 3-36.
SMMU_CB*n*_PMCEID0-1	-	-	0x00F20	See *Performance Monitoring registers* on page 3-36.
SMMU_CB*n*_PMCNTENSET	-	-	0x00F40	See *Performance Monitoring registers* on page 3-36.
SMMU_CB*n*_PMCNTENCLR	-	-	0x00F44	See *Performance Monitoring registers* on page 3-36.
SMMU_CB*n*_PMINTENSET	-	-	0x00F48	See *Performance Monitoring registers* on page 3-36.
SMMU_CB*n*_PMINTENCLR	-	-	0x00F4C	See *Performance Monitoring registers* on page 3-36.
SMMU_CB*n*_PMOVSRCLR	-	-	0x00F54	See *Performance Monitoring registers* on page 3-36.
SMMU_CB*n*_PMOVSRSET	-	-	0x00F50	See *Performance Monitoring registers* on page 3-36.
SMMU_CB*n*_PMAUTHSTATUS	-	-	0x00FB8	See *Performance Monitoring registers* on page 3-36.

a. Follow the same format as *Global Register Space 0* on page 3-10 describes.

3.4 Global Register Space 0

The MMU-400 Global Register Space 0 provides high-level control of the MMU-400 resources and maps device transactions to translation context banks. It contains the following registers:

- *Secure Configuration Register 0*.
- *Auxiliary Configuration Register* on page 3-16.
- *Identification registers* on page 3-17.
- *Debug registers* on page 3-23.
- *Secure Alias to Non-secure Configuration Register 0* on page 3-27.
- *Secure Alias to Non-secure Auxiliary Configuration Register* on page 3-27.
- *Stream Match Registers* on page 3-27.
- *Stream to Context registers* on page 3-29.

3.4.1 Secure Configuration Register 0

The characteristics of the Secure Configuration Register 0 are:

Purpose The Secure Configuration Register, 0, SMMU_SCR0, provides top-level control of the MMU-400 that is only accessible by Secure accesses.

Note

The Non-secure register, SMMU_CR0, does not provide full top-level control of the MMU-400 for Secure transactions. In implementations that support security extensions, some SMMU_CR0 fields only apply to Non-secure transactions.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes See Table 3-1 on page 3-4.

Figure 3-2 on page 3-11 shows the bit assignments.

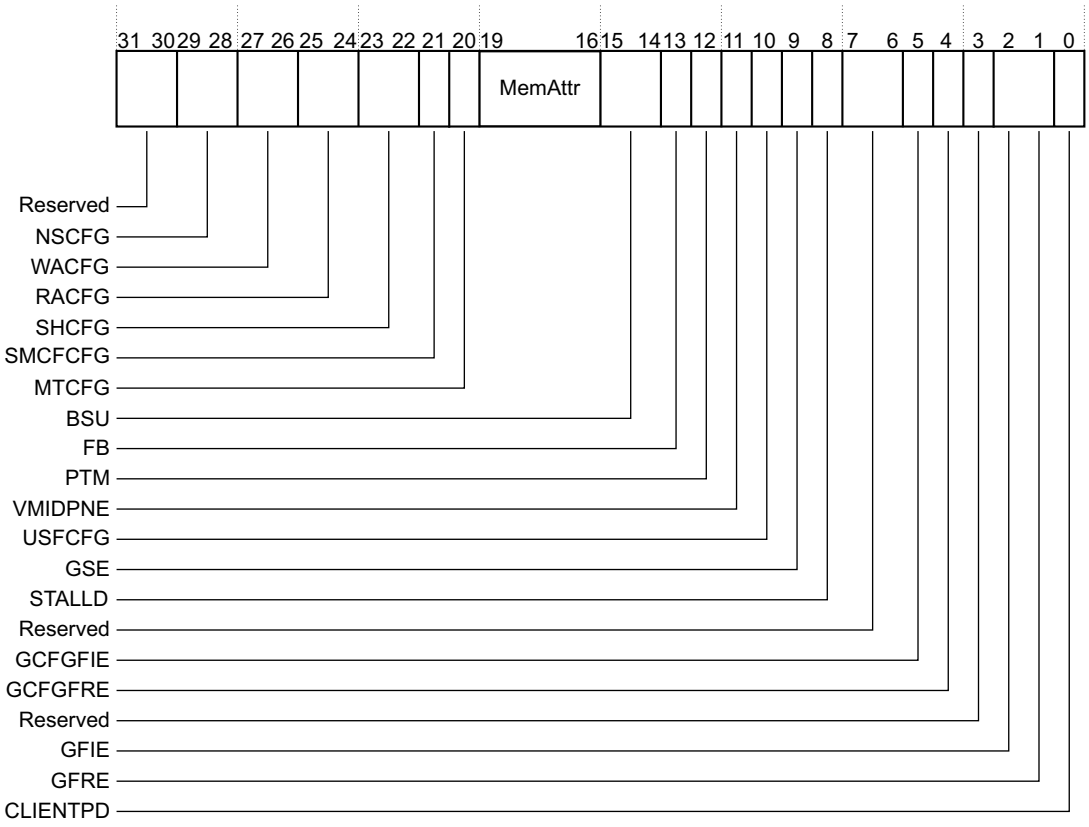


Figure 3-2 Secure Configuration Register 0 bit assignments

Table 3-8 shows the bit assignments.

Table 3-8 Secure Configuration Register 0 bit assignments

Bits	Name	Reset value	Description
[31:30]	Reserved	-	Reserved.
[29:28]	NSCFG	-	Non-secure Configuration. The encodings of this field are: 0b00 Use APROTNS as presented with the transaction. 0b01 Reserved. 0b10 Secure. 0b11 Non-secure. **Note** • This field is only present in SMMU_SCR0. In SMMU_CR0, it is reserved. • This field applies to the processing of Secure transactions that bypass the MMU-400 stream mapping process. This bypass condition can occur if: — SMMU_SCR0.CLIENTPD has the value 1. — The transaction does not match in the stream mapping table and SMMU_SCR0.USFCFG has the value 0.
[27:26]	WACFG	-	Write Allocate Configuration. Controls the allocation hint for write accesses. The encodings of this field are: 0b00 Use allocation attributes as presented with the transaction. 0b01 Reserved. 0b10 Allocate. 0b11 No allocate. **Note** This field applies to transactions that bypass the MMU-400 stream mapping process. This bypass condition can occur if: • SMMU_CR0.CLIENTPD has the value 1. • The transaction does not match in the stream mapping table, and SMMU_CR0.USFCFG has the value 0.
[25:24]	RACFG	-	Read Allocate Configuration. Controls the allocation hint for read accesses. The encodings of this field are: 0b00 Use allocation attributes as presented with the transaction. 0b01 Reserved. 0b10 Allocate. 0b11 No allocate. **Note** This field applies to transactions that bypass the MMU-400 stream mapping process. This bypass condition can occur if: • SMMU_CR0.CLIENTPD has the value 1. • The transaction does not match in the stream mapping table, and SMMU_CR0.USFCFG has the value 0.

Table 3-8 Secure Configuration Register 0 bit assignments (continued)

Bits	Name	Reset value	Description
[23:22]	SHCFG	-	Shared Configuration. The encodings of this field are: `0b00` Use shareable attribute as presented with the transaction. `0b01` Outer-shareable. `0b10` Inner-shareable. `0b11` Non-shareable. **Note** This field applies to transactions that bypass the MMU-400 stream mapping process. This bypass condition can occur if: • SMMU_CR0.CLIENTPD has the value 1. • The transaction does not match in the stream mapping table, and SMMU_CR0.USFCFG has the value 0.
[21]	SMCFCFG	-	Stream Match Conflict Fault Configuration. Controls the handling of client transactions that are identified as matching multiple entries in the stream mapping table. The possible values of this are: `0` Permit passes through. `1` Raise a stream match conflict fault. **Note** The MMU-400 treats this bit as RAO/WI. The MMU-400 does not guarantee detection of all occurrences of the stream match conflict fault, see the *System Memory Management Unit Architecture Specification*.
[20]	MTCFG	0	Memory Type Configuration. `0` Use memory attributes as presented with the transaction. `1` Use MemAttr field for memory attributes. **Note** This field applies to the processing of Non-secure transactions that bypass the MMU stream mapping process. This bypass condition can occur if: • SMMU_CR0.CLIENTPD has the value 1. • The transaction does not match in the stream mapping table, and SMMU_CR0.USFCFG has the value 0.
[19:16]	MemAttr	-	Memory Attributes. Memory attributes can be overlaid if SMMU_CR0.MTCFG has the value 1.

Table 3-8 Secure Configuration Register 0 bit assignments (continued)

Bits	Name	Reset value	Description
[15:14]	BSU	-	Barrier Shareability Upgrade. Upgrades the required shareability domain of barriers issued by client devices that are not mapped to a translation context, by setting the minimum shareability domain that is applied to any barrier. The encodings of this field are: `0b00` No effect. `0b01` Inner shareable. `0b10` Outer shareable. `0b11` Full system. **Note** The MMU-400 supports the BSU only in ACE-Lite configurations. This field only applies to barriers that are received by the MMU-400.
[13]	FB	-	Force Broadcast. Force broadcast of TLB, branch predictor, and instruction cache maintenance operations. This field affects the TLB maintenance, BPIALL, and ICIALLU operations. If FB is set to 1, then the affected operations are modified to the equivalent broadcast variant in the inner shareable domain. The possible values of this bit are: `0` Process affected operations as presented. `1` Upgrade affected operations to be broadcast within the inner shareable domain. **Note** This field applies to the processing of transactions that bypass the MMU-400 stream mapping process. For Non-secure transactions, this bypass condition can occur if: • SMMU_CR0.CLIENTPD has the value 1. • The transaction does not match in the stream mapping table and SMMU_CR0.USFCFG has the value 0. A similar set of conditions exist for Secure transaction bypass.
[12]	PTM	0	Private TLB Maintenance. The possible values of this bit are: `0` As indicated by SMMU_IDR0.PTM. If it is supported in the implementation, then the MMU-400 must participate in the Broadcast TLB maintenance with the wider system. `1` The MMU-400 TLBs are managed privately from the wider system and it is not necessary to respond to the Broadcast TLB maintenance operations. The PTM field is a hint. A broadcast TLB invalidate operation is still can affect cached translation in the MMU-400, and can apply to all unlocked entries. In implementations that support security extensions, this field only has an effect on Non-secure owned TLB entries. An equivalent functionality for Secure owned TLB entries is provided in SMMU_SCR0.PTM.

Table 3-8 Secure Configuration Register 0 bit assignments (continued)

Bits	Name	Reset value	Description
[11]	VMIDPNE	-	VMID Private Namespace Enable. The possible values of this bit are: 0 The MMU-400 VMID values are coordinated with the wider system. 1 The MMU-400 VMID values are a private namespace and are not coordinated with the wider system. If VMIDPNE has the value 1, broadcast TLB Invalidate operations that specify a VMID value do not have to apply to cached translations in the MMU-400. This field is a hint. A broadcast TLB Invalidate operation is still can affect cached translations in the MMU-400 and can apply to all unlocked entries. **Note** This field is reserved in SMMU_SCR0.
[10]	USFCFG	0	Unidentified Stream Fault Configuration. The possible values of this bit are: 0 Permit transactions that do not match any entries in the stream mapping table to pass through. 1 Raise an unidentified stream fault on transactions that do not match any entries in the stream mapping table.
[9]	GSE	0	Global Stall Enable. The possible values of this bit are: 0 Do not enforce global stalling across contexts. 1 Enforce global stalling across contexts. This bit is RAZ/WI.
[8]	STALLD	0	Stall Disable. The possible values of this bit are: 0 Enable per-context stalling on context faults. 1 Disable per-context stalling on context faults. This bit behaves as RAO/WI. In implementations that support security extensions, SMMU_CR0.STALLD must apply to a Non-secure translation context bank. It can optionally apply to a Secure translation context bank, and can affect SMMU_SCR0.GSE.
[7:6]	Reserved	-	Reserved.
[5]	GCFGFIE	0	Global Configuration Fault Interrupt Enable. The possible values of this bit are: 0 Do not raise an interrupt on a global configuration fault. 1 Raise an interrupt on a global configuration fault.
[4]	GCFGFRE	0	Global Configuration Fault Report Enable. The possible values of this bit are: 0 Do not return an abort on a global configuration fault. 1 Return an abort on a global configuration fault.
[3]	Reserved	-	Reserved.

Table 3-8 Secure Configuration Register 0 bit assignments (continued)

Bits	Name	Reset value	Description
[2]	GFIE	0	Global Fault Interrupt Enable. The possible values of this bit are: 0 Do not raise an interrupt on a global fault. 1 Raise an interrupt on a global fault.
[1]	GFRE	0	Global Fault Report Enable. The possible values of this bit are: 0 Do not return an abort on a global fault. 1 Return an abort on a global fault.
[0]	CLIENTPD	1	Client Port Disable. The possible values of this bit are: 0 The MMU-400 client accesses are subject to translation, access control, and attribute generation. 1 The MMU-400 client accesses bypass translation, access control, and attribute generation.

3.4.2 Auxiliary Configuration Register

The characteristics of the Auxiliary Configuration Registers are:

Purpose The Auxiliary Configuration Registers, SMMU_ACR and SMMU_SACR, are defined as shown in Table 3-9 on page 3-17 .

Configuration Available in all MMU-400 configurations.

Usage constraints The WC2EN, WC1EN, and PREFETCHEN bits are Non-secure only. Other bits are banked with security.

Attributes See *Global space 0 registers summary* on page 3-4.

Figure 3-3 shows the bit assignments.

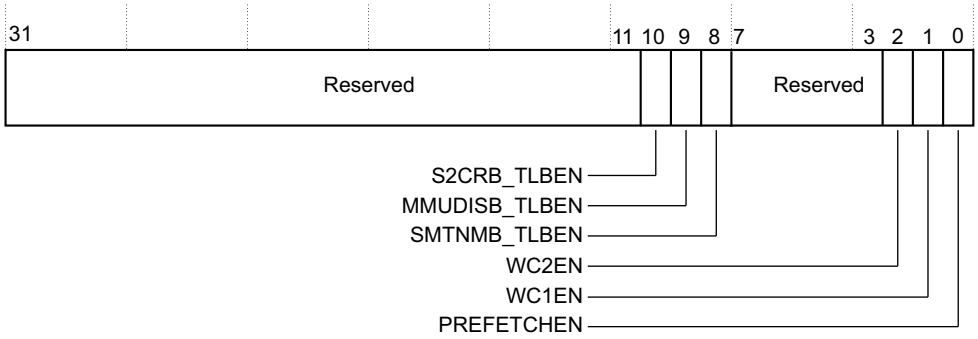


Figure 3-3 Auxiliary Configuration Register bit assignments

Note

The bits[0:2] are valid only for the ACR register, whereas the bits[8:10] are valid for both the ACR and SACR registers.

Table 3-9 shows the bit assignments.

Table 3-9 Auxiliary Configuration Register bit assignments

Bits	Name	Description
[31:11]	Reserved	Reserved.
[10]	S2CRB_TLBEN	Stream to Context Register Bypass TLB Enabled. The possible values of this bit are: 0 Do not update the TLB with the S2CR bypassed transaction details. 1 Update the TLB with the S2CR bypassed transaction details.
[9]	MMUDISB_TLBEN	MMU Disabled Bypass TLB Enable. The MMU-400 caches in the TLB the attribute information for transactions that have been allocated a context, but the SCTRL.M bit of that context was set to 0. This caching saves six clock cycles for handling of such transactions as a minimum, but could save many more, depending on how busy the SMMU is. The possible values of this bit are: 0 Do not update the TLB with the MMU-400 disabled transaction details. 1 Update the TLB with the MMU-400 disabled transaction details.
[8]	SMTNMB_TLBEN	Stream Match Table No Match TLB Enabled. The possible values of this bit are: 0 Do not update the TLB with the stream match table no match bypassed transaction details. 1 Update the TLB with the stream match table no match bypassed transaction details.
[7:3]	Reserved	Reserved.
[2]	WC2EN	Walk Cache 2 Enable. The MMU-400 caches the L2 Page Table Walk in the walk cache 2. The possible values of this bit are: 0 Disables the walk cache 2 functionality. 1 Enables the walk cache 2 functionality.
[1]	WC1EN	Walk Cache 1 Enable. The MMU-400 caches the L1 Page Table Walk in the walk cache 1. The possible values of this bit are: 0 Disables the walk cache 1 functionality. 1 Enables the walk cache 1 functionality.
[0]	PREFETCHEN	Pre-fetch buffer Enable. The MMU-400 prefetches the next page table entry while doing a L3 Page Table Walk. The possible values of this bit are: 0 Disables the pre-fetch buffer. 1 Enables the pre-fetch buffer.

3.4.3 Identification registers

The characteristics of the Identification Register are:

Purpose The identification registers, SMMU_IDR and SMMU_SIDR, provide information on the capability of the MMU-400. This section describes the following Identification registers:

- *Identification Register 0* on page 3-18.
- *Identification Register 1* on page 3-20.
- *Identification Register 2* on page 3-22.
- *Identification Register 7* on page 3-22.

The MMU-400 provides facilities to permit the Secure software to reserve some MMU-400 resources for its own use. See *Identification Register 0*.

The Non-secure versions of the SMMU_IDR registers report the number of resources taking into account the number reserved by the Secure software, that is, a number potentially lower than the number of physically-implemented resources.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes *Global space 0 registers summary* on page 3-4.

Identification Register 0

Figure 3-4 shows the bit assignments.

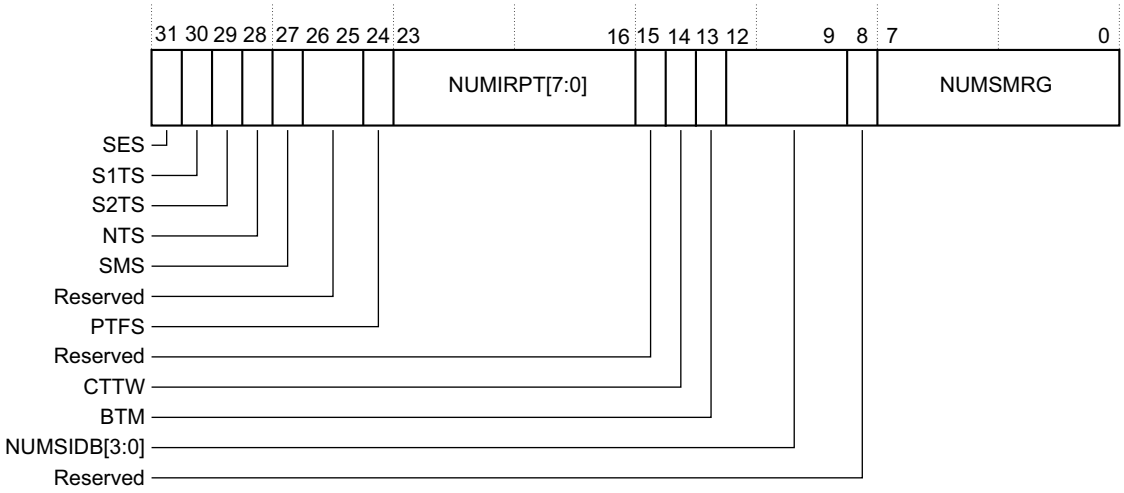


Figure 3-4 Identification Register 0 bit assignments

Table 3-10 shows the bit assignments.

Table 3-10 Identification Register 0 bit assignments

Bits	Name	Reset value	Description
[31]	SES	1	Security Extension Support.
[30]	S1TS	0	Stage 1 translation support. This is not supported in the MMU-400.
[29]	S2TS	1	Stage 2 Translation Support. The possible values of this bit are: `0` Stage 2 translation not supported. `1` Stage 2 translation supported. This field only applies to Non-secure client transactions.
[28]	NTS	0	Nested Translation Support. The possible values of this bit are: `0` Nested translation not supported. `1` Nested translation supported. This field only applies to Non-secure client transactions.

Table 3-10 Identification Register 0 bit assignments (continued)

Bits	Name	Reset value	Description
[27]	SMS	1	Stream Match Support. The possible values of this bit are: 0 Stream match register functionality not present. 1 Stream match register functionality present.
[26:25]	Reserved	-	Reserved.
[24]	PTFS	1	Page Table Format Support. The possible values of this bit are: 0 VMSA v7 and LPAE formats supported. 1 LPAE format supported.
[23:16]	NUMIRPT[7:0]	1	Number of Implementation Context Interrupts. NUMIRPT[7:0] indicates the number of context fault interrupts that the MMU-400 supports. It supports only one context interrupt. **Note** Permitting the value 0 anticipates implementations that provide no context banks. **Interaction with security extensions** In implementations that support security extensions, the SMMU_IDR0 version of this field reflects the configured value in SMMU_SCR1.NSNUMIRPTO. **Note** When the Secure software provides one or more context banks to the Non-secure software, it must also provide at least one use of context interrupt.
[15]	Reserved	-	Reserved.
[14]	CTTW	tie-off to **cfg_cttw**	Coherent Translation Table Walk. The possible values of this bit are: 0 Coherent translation table walk not supported. 1 Coherent translation table walk supported.
[13]	BTM	When you select: **AXI3/4** Then the reset value is 0 **ACE-Lite** Then the reset value is 1	Broadcast TLB Maintenance. The possible values of this bit are: 0 Broadcast TLB maintenance not supported. 1 Broadcast TLB maintenance supported.

Table 3-10 Identification Register 0 bit assignments (continued)

Bits	Name	Reset value	Description
[12:9]	NUMSIDB[3:0]	Configured StreamID width	Number of StreamID bits: NUMSIDB[3:0] indicates the number of StreamID bits that are implemented. The valid range is 0-15. **Note** You can anticipate a 0-bit StreamID, where the source of a single Stream ID has a dedicated MMU-400.
[8]	Reserved	-	Reserved.
[7:0]	NUMSMRG[7:0]	Configured number of stream mapping registers	Number of Stream Mapping Register Groups. NUMSMRG[7:0] indicates the number of entries in the stream mapping table. In implementations that support the stream match function, this field has a value of greater than or equal to 1. **Interaction with security extensions** In implementations that support security extensions, the SMMU_IDR0 version of this field reflects the configured value in SMMU_SCR1.NSNUMSMRGO. **Note** In implementations that support the stream match function, the Secure software must provide the Non-secure software with the use of at least one SMR group.

Identification Register 1

Figure 3-5 shows the bit assignments.

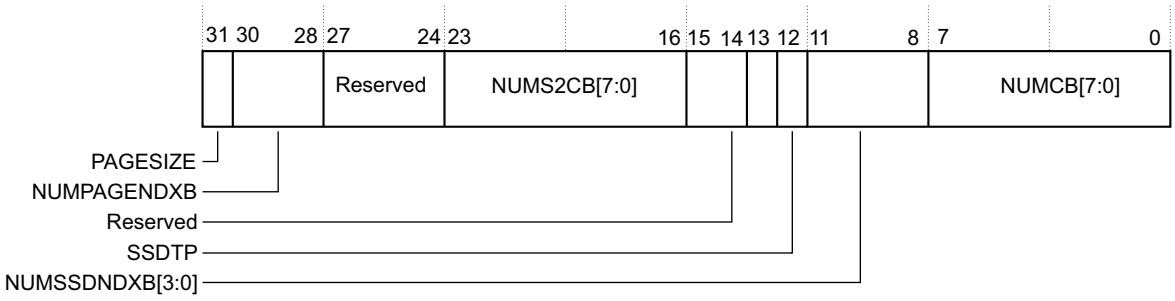


Figure 3-5 Identification Register 1 bit assignments

Table 3-11 shows the bit assignments.

Table 3-11 Identification Register 1 bit assignments

Bits	Name	Reset value	Description
[31]	PAGESIZE	0	The MMU-400 page size. Indicates the size of each page in the MMU-400 register map. The possible values of this bit are: 0 The PAGESIZE is 4KB. 1 The PAGESIZE is 64KB.
[30:28]	NUMPAGENDXB	0b010	The MMU-400 number of page index bits. Indicates the number of PAGESIZE pages occupying the global address space or translation context address space: $NUMPAGE = 2^{(NUMPAGENDXB+1)}$
[27:24]	Reserved	-	Reserved.
[23:16]	NUMS2CB[7:0]	Configured number of context banks	Number of stage 2 context banks. Indicates the number of context banks that only support the stage 2 translation. This field is validated by SMMU_IDR0.S2TS.
[15]	SMCD	0	Stream match conflict detection. The possible values of this bit are: 0 Not all stream match conflicts are guaranteed to be detected. 1 All stream match conflicts are guaranteed to be detected. See the *ARM® System Memory Management Unit Architecture Specification*.
[14:13]	Reserved	-	Reserved.
[12]	SSDTP	As configured	Secure status determination table present. The possible values of this bit are: 0 Secure status determination address space UNK/WI. 1 Secure status determination address space populated. **Note** This field is RAZ for Non-secure reads of SMMU_IDR1.
[11:8]	NUMSSDNDXB[3:0]	4'hF	Number of SSD_Index bits. Indicates the number of SSD_Index bits used to index into the Secure status determination Table. This field is only valid if SSDTP is HIGH, otherwise this field is reserved. **Note** This field is RAZ for Non-secure reads of SMMU_IDR1.
[7:0]	NUMCB[7:0]	Configured number of context banks	Number of context banks. Indicates the total number of translation context banks that are implemented. See *Translation Context Bank registers* on page 3-50. The value reported in NUMCB includes translation context banks that only support the stage 2 translation. **Interaction with security extensions** In implementations that support security extensions, the Non-secure reads of SMMU_IDR1 reflect the configured value in SMMU_SCR1.NSNUMCBO.

Identification Register 2

Figure 3-6 shows the bit assignments.

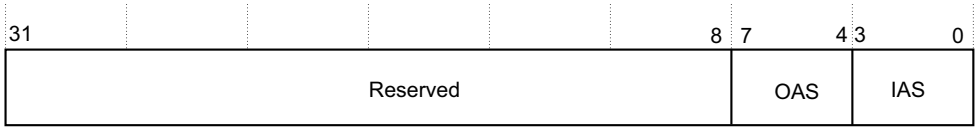


Figure 3-6 Identification Register 2 bit assignments

Table 3-12 shows the bit assignments.

Table 3-12 Identification Register 2 bit assignments

Bits	Name	Reset value	Description
[31:8]	Reserved	-	Reserved.
[7:4]	OAS	0b0010	Output address size. The encodings of this field are: 0b0000 32 bits output address size. 0b0001 36 bits output address size. 0b0010 40 bits output address size. All other encodings are reserved.
[3:0]	IAS	0b0010	Input address size. The encodings of this field are: 0b0000 32 bits input address size. 0b0001 36 bits input address size. 0b0010 40 bits input address size. All other encodings are reserved.

Identification Register 7

Figure 3-7 shows the bit assignments.

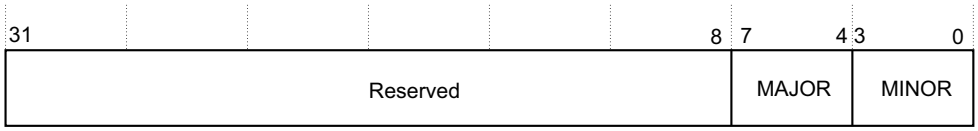


Figure 3-7 Identification Register 7 bit assignments

Table 3-13 shows the bit assignments.

Table 3-13 Identification Register 7 bit assignments

Bits	Name	Reset value	Description
[31:8]	Reserved	-	Reserved.
[7:4]	MAJOR	0	The major part of the implementation version number.
[3:0]	MINOR	0	The minor part of the implementation version number.

3.4.4 Debug registers

Purpose

The MMU-400 supports TLB visibility by providing a read pointer register and a read data register to read the TLB entries. The read pointer register is initialized at reset to 0, and is auto-incremented by one word, that is, four bytes, whenever a read to the TLB data register happens.

When a read to the TLB data register happens, the TLB data present at the read pointer register is read and returned on the APB.

For the read pointer register, the lower two bits are RAZ/WI. This is to ensure that the address for a debug TLB fetch is always word aligned. If the read pointer register is written with an address value that is out of bounds of the TLB then this results in APB accesses returning an error when the corresponding read is made to the read data register.

See *Global space 0 registers summary* on page 3-4.

It is possible to read the particular entry in the TLB by programming the read pointer register. When the read data register is read, the TLB entry pointed to by the read pointer is read back and the read pointer is auto incremented. If the read data register is read again then the next entry in the TLB is read back.

ARM recommends that TLB reads occur when there are no outstanding transactions. If TLB reads occur in conjunction with transactions, then the TLB read can return data before or after it has been updated.

It contains the following registers:

- *SMMU_DBGRPTR, Debug Read Pointer Register*.
- *SMMU_DBGRDATA, Debug Read Data Register* on page 3-24.

Configuration Available in all MMU-400 configurations.

Usage constraints Only Secure access is possible.

Attributes *Global space 0 registers summary* on page 3-4.

SMMU_DBGRPTR, Debug Read Pointer Register

For the MMU-400, bits[31:16] are always 0 unless written specifically by an APB access.

Figure 3-8 shows the bit assignments.

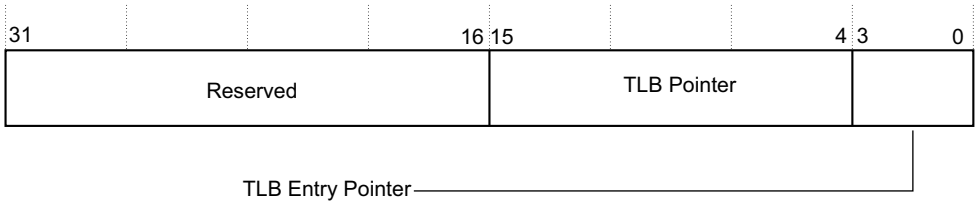


Figure 3-8 Debug Read Pointer Register bit assignments

Table 3-14 shows the bit assignments.

Table 3-14 Debug Read Pointer Register bit assignments

Bits	Name	Description
[31:16]	Reserved	Reserved.
[15:4]	TLB Pointer	Pointing to a specific TLB Entry.
[3:0]	TLB Entry Pointer	Words with in TLB entry.

SMMU_DBGRDATA, Debug Read Data Register

The following tables show the bit assignments for words 1-5:

- Table 3-15 on page 3-25.
- Table 3-16 on page 3-25.
- Table 3-17 on page 3-26.
- Table 3-18 on page 3-26.

- Table 3-19 on page 3-27.

Table 3-15 Debug Read Data Register data format, word 1

Bits	Width	Description
[31:4]	28	Virtual Address used for address lookup.
[3:2]	2	TLB_ENTRY_VALID. This bit field specifies whether the TLB entry is valid. This bit can have one of the following values: `0b00` Word which is not first or last of the TLB entry. `0b01` First word for that TLB entry. `0b10` Last word for that TLB entry. `0b11` First word for that TLB.
[1]	1	TLB_POINTER_VALID. This bit specifies whether the TLB pointer is valid.This bit can have one of the following values: `0b0` Pointer is valid. `0b1` Pointer is invalid.
[0]	1	TLB_WORD_INFO. This bit specifies whether the TLB word information is valid. This bit can have one of the following values: 0b0 Valid. 0b1 Invalid. The possible values of this bit are: `0` Entry is valid. `1` Entry is invalid.

Table 3-16 Debug Read Data Register data format, word 2

Bits	Width	Description
[31:30]	2	Reserved.
[29:28]	2	PRIVCFG, Privilege Configuration. See the *ARM® System Memory Management Unit Architecture Specification*.
[27:26]	2	INSTCFG, Instruction Configuration. See the *ARM® System Memory Management Unit Architecture Specification*.
[25:24]	2	NSCFG, Non-secure Configuration. See the *ARM® System Memory Management Unit Architecture Specification*.
[23:17]	7	Memory Attributes.
[16:14]	3	SHCFG, Shareability Configuration. See the *ARM® System Memory Management Unit Architecture Specification*.
[13:12]	2	RACFG, Read Allocate Configuration. See the *ARM® System Memory Management Unit Architecture Specification*.
[11:10]	2	WACFG, Write Allocate Configuration. See the *ARM® System Memory Management Unit Architecture Specification*.
[9:8]	2	BSU, Barrier Shareability Upgrade. See the *ARM® System Memory Management Unit Architecture Specification*.
[7]	1	Reserved.

Table 3-16 Debug Read Data Register data format, word 2 (continued)

Bits	Width	Description
[6:4]	3	Page Size. The encodings of this field are: 0b000 4KB. 0b001 Reserved. 0b010 Reserved. 0b011 2MB. 0b100 Reserved. 0b101 1GB. 0b110 Reserved. 0b111 Reserved.
[3:2]	2	TLB_ENTRY_VALID. *Debug Read Data Register data format, word 1* on page 3-25.
[1]	1	TLB_POINTER_VALID. *Debug Read Data Register data format, word 1* on page 3-25.
[0]	1	*Debug Read Data Register data format, word 1* on page 3-25.

Table 3-17 Debug Read Data Register data format, word 3

Bits	Width	Description
[31:17]	15	StreamID.
[16]	1	Reserved.
[15:8]	8	Translation Context Index.
[7:5]	3	Reserved.
[4]	1	Non-secure state, security status of the master that initiated this transaction.
[3:2]	2	TLB_ENTRY_VALID. *Debug Read Data Register data format, word 1* on page 3-25.
[1]	1	TLB_POINTER_VALID. *Debug Read Data Register data format, word 1* on page 3-25.
[0]	1	*Debug Read Data Register data format, word 1* on page 3-25.

Table 3-18 Debug Read Data Register data format, word 4

Bits	Width	Description
[31:17]	15	StreamID Mask.
[16:13]	4	Reserved.
[12:11]	2	Entry Type. The encodings of this field are: b00 Translation is enabled. b01 Translation is disabled. b10 S2CR bypass as programmed in the S2CR Register. b11 USFCFG bypass. The CR0.USFCFG bit is set.
[10]	1	16 contiguous entry hint.
[9]	1	PXN, Privilege Execute Never. See the *ARM® System Memory Management Unit Architecture Specification*.

Table 3-18 Debug Read Data Register data format, word 4 (continued)

Bits	Width	Description
[8]	1	XN, Executer Never. See the *ARM® System Memory Management Unit Architecture Specification*.
[7:5]	3	HAP. See the *ARM® System Memory Management Unit Architecture Specification*.
[4]	1	AFE. See the *ARM® System Memory Management Unit Architecture Specification*.
[3:2]	2	TLB_ENTRY_VALID. *Debug Read Data Register data format, word 1* on page 3-25.
[1]	1	TLB_POINTER_VALID. *Debug Read Data Register data format, word 1* on page 3-25.
[0]	1	*Debug Read Data Register data format, word 1* on page 3-25.

Table 3-19 Debug Read Data Register data format, word 5

Bits	Width	Description
[31:4]	28	Physical address.
[3:2]	2	TLB_ENTRY_VALID. *Debug Read Data Register data format, word 1* on page 3-25.
[1]	1	Reserved.
[0]	1	*Debug Read Data Register data format, word 1* on page 3-25.

3.4.5 Secure Alias to Non-secure Configuration Register 0

Purpose The Secure Alias to Non-secure Configuration Register 0 is accessed by Secure transactions. See *Secure Configuration Register 0* on page 3-10.

Configuration Available in all MMU-400 configurations.

Usage constraints GSE is RAZ/WI. STALLD and SMCFCFG are RAO/WI.

Attributes *Global space 0 registers summary* on page 3-4.

3.4.6 Secure Alias to Non-secure Auxiliary Configuration Register

Purpose The Secure Alias to Non-secure Auxiliary Configuration Register is accessed by Secure transactions. See *Auxiliary Configuration Register* on page 3-16.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes *Global space 0 registers summary* on page 3-4.

3.4.7 Stream Match Registers

The characteristics of the Stream Match Registers are:

Purpose The Stream Match registers, SMMU_SMR*n*, match a transaction with a particular context mapping register group.

The MMU-400 supports StreamID matching, and all these registers are treated as R/W.

The Stream Match Registers form a table that is searched to find a match for a transaction StreamID. The StreamID uniquely identifies the originator of a transaction, and you might commonly derive the StreamID from identifier information conveyed on the bus interconnect:

- NS.
- RnW.
- ID.

You can identify a number of StreamID values as belonging to the same context. This permits the sharing of the state describing that context. Mapping multiple StreamID encodings to the same context is achieved using multiple Stream Match Register entries, or the mask facilities available in the Stream Match Register encoding.

An active StreamID, that is, a stream that has transactions in progress or that is issuing transactions, can match at most one entry in the Stream Match Register table. If the StreamID of a transaction matches multiple stream mapping table entries, the following action is taken:

- The multiple match condition is trapped by the MMU-400. The transaction is terminated at the MMU-400, and an SMR multiple match fault is recorded in the SMMU_GFSR.

The memory or MMU-400 state that is not accessible through the selected matching stream mapping table entry must remain unaffected.

The Stream Match Register table can have multiple entries matching the same StreamID value during configuration, providing software has the necessary precautions before configuration takes effect. For example:

- Disable the stream source and ensure that no outstanding transactions from that source are in progress.
- Disable one or more of the SMMU_SMRn table entries using the corresponding SMMU_SMRn.VALID bit.
- Disable the MMU-400 completely with the global MMU-400 enable.

The number of ID and MASK bits is configured as described in *Configurable options* on page 1-6. Unimplemented bits are RAZ/WI. An implementation must provide the same number of ID and MASK bits for every implemented Stream Match Register.

The number of SMMU_SMRn registers actually present is configured as described in *Configurable options* on page 1-6. The unimplemented registers or those reserved by Secure software and so not visible to Non-secure access are RAZ/WI.

Configuration The width of MASK and ID fields is equal to the configured StreamID width.

Usage constraints There are no usage constraints.

Attributes See *Global space 0 registers summary* on page 3-4.

Figure 3-9 on page 3-29 shows the bit assignments.

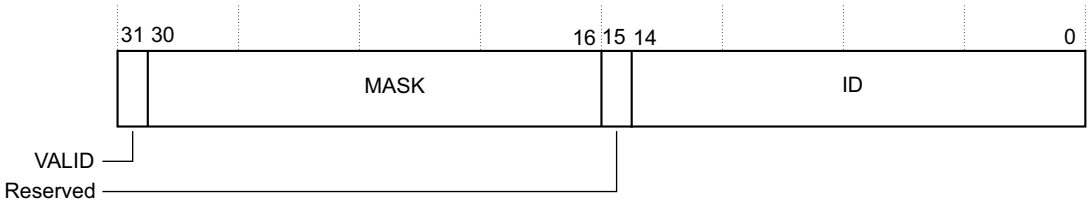


Figure 3-9 Stream Match registers bit assignments

Table 3-20 shows the bit assignments.

Table 3-20 Stream Match registers bit assignments

Bits	Name	Description
[31]	VALID	Entry included or not in stream mapping table search: `1` Entry included in stream mapping table search. `0` Entry not included in stream mapping table search.
[30:16]	MASK	Mask. Identifies if the StreamID is ignored: **MASK[i]==**`1` ID[i] is ignored. **MASK[i]==**`0` ID[i] is relevant to match.
[15]	Reserved	Reserved.
[14:0]	ID	StreamID to match.

3.4.8 Stream to Context registers

The characteristics of the Stream to Context Registers are:

Purpose Specifies an initial context for processing a transaction, where that transaction matches the stream mapping group that this register belongs to.

The number of SMMU_S2CRn registers is configured as described in *Configurable options* on page 1-6. The unimplemented registers are RAZ/WI.

The format of the SMMU_S2CRn registers depends on the state of its TYPE field. See the *ARM® System Memory Management Unit Architecture Specification*.

Configuration The width of CBNDX field depends on the number of contexts:

1 or 2 contexts 1 bit.

3 or 4 contexts 2 bits.

5, 6, 7, and 8 contexts 3 bits.

Usage constraints The VMID field in bypass mode is RAZ/WI.

Attributes See *Global space 0 registers summary* on page 3-4.

3.5 Global Register Space 1

The MMU-400 Global register space 1 provides high-level control of the MMU-400 resources and maps device transactions to translation context banks. It contains the following registers:

- *Context Bank Fault Restricted Syndrome Register A*.
- *Context Bank Attribute Register* on page 3-31.

3.5.1 Context Bank Fault Restricted Syndrome Register A

The characteristics of the Context Bank Fault Restricted Syndrome Register A are:

Purpose The Context Bank Fault Restricted Syndrome register A, SMMU_CBFRSYNRA*n*, holds fault syndrome information related to the access that caused an exception in the associated translation context bank. These registers are located in the MMU-400 global address space, as the information contained in them has the potential to be a virtualization hole, if revealed within the SMMU_CBn_FSYNR registers.

The number of registers implemented is discovered by reading the SMMU_IDR.NUMCB field.

A context bank of index *n* is associated with a Context Bank Fault Restricted Syndrome A Register of index *n*.

In an implementation that supports security extensions, the Secure software that reserves translation context banks using the SMMU_SCR1.NSNUMCB0 field also reserves the Context Bank Fault Restricted Syndrome A registers associated with the translation context banks that have been reserved.

The number of Context Bank Fault Restricted Syndrome A registers visible to the Non-secure software is adjusted accordingly.

Configuration The width of StreamID and SSD_Index fields are as configured.

Usage constraints There are no usage constraints.

Attribute *Global space 1 registers summary* on page 3-6.

Figure 3-10 shows the bit assignments.

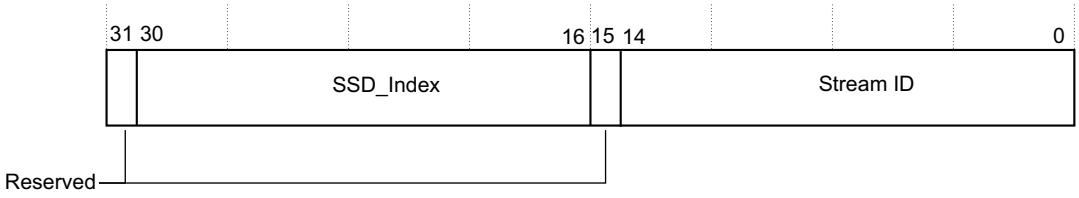


Figure 3-10 Context Bank Restricted Syndrome Register bit assignments

Table 3-21 shows the bit assignments.

Table 3-21 Context Bank Restricted Syndrome Register bit assignments

Bits	Name	Description
[31]	Reserved	Reserved.
[30:16]	SSD_Index	SSD_Index of transaction causing fault. **Note** This field is only accessible to Secure configuration accesses. Non-secure configuration accesses treat this field as RAZ/WI.
[15]	Reserved	Reserved.
[14:0]	StreamID	The StreamID of the transaction that caused the fault.

3.5.2 Context Bank Attribute Register

The characteristics of the Context Bank Attribute Register are:

Purpose The Context Bank Attribute register, SMMU_CBAR*n*, specifies additional configuration attributes for a translation context bank.

The number of registers implemented is discovered by reading the SMMU_IDR.NUMCB field.

A context bank of index *n*, is associated with a Context Bank Attribute Register of index, *n*.

There are a number of SMMU_CBAR*n* encoding formats, dependent on how the TYPE field is configured as Table 3-22 shows.

Table 3-22 Context Bank Attribute Register

SMMU_CBARn[TYPE]	SMMU_CBARn Format	Description
`0b00`	Stage 2 Context	*Stage 2 Context, TYPE==0b00* on page 3-32
`0b01`	Reserved	Reserved
`0b10`	Reserved	Reserved
`0b11`	Reserved	Reserved

Note

The SMMU_CBAR*n* registers associated with a translation context bank that only supports stage 2 translation have their TYPE field fixed at `0b00`, and format selected accordingly.

For the MMU-400, the IRPTNDX bit is RO.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes *Global space 1 registers summary* on page 3-6.

Stage 2 Context, TYPE==0b00

Figure 3-11 shows the stage 2 context, TYPE==0b00 format that configures the associated translation context bank to provide stage 2 translation.

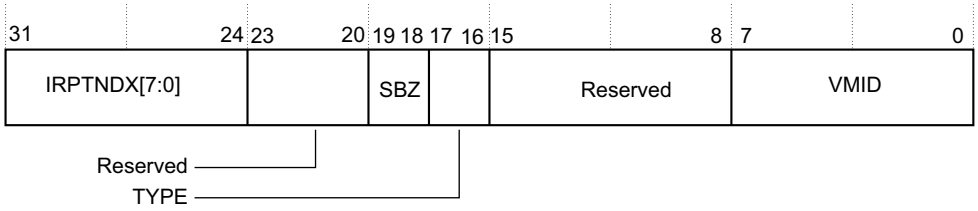


Figure 3-11 Stage 2 Context, TYPE==0b00 Register bit assignments

Table 3-23 shows the stage 2 context, TYPE==0b00 format that configures the associated translation context bank to provide stage 2 translation.

Table 3-23 Stage 2 Context, TYPE==0b00 Register bit assignments

Bits	Name	Description
[31:24]	IRPTNDX[7:0]	Interrupt Index. The context interrupt number to assert in the event of an interrupt raising a fault in the associated translation context bank. This bit is RO.
[23:20]	Reserved	Reserved.
[19:18]	SBZ	-
[17:16]	TYPE	CBAR*n* Type. Indicates the format of the remaining fields within this register. This field behaves as RAZ/WI.
[15:8]	Reserved	Reserved.
[7:0]	VMID	The Virtual Machine Identifier to be associated with the translation context bank. **Note** For the stage 2 context format, this field is only used when the associated stage 2 translation context bank is the first context that is specified in an SMMU-S2CRn register is used.

3.6 Integration registers

This section describes the integration registers for the MMU-400. It contains the following sections:

- *Integration Enable Register*.
- *Integration Test Input Register* on page 3-34.
- *Integration Test Output Register* on page 3-34.

3.6.1 Integration Enable Register

The characteristics of the ITEN Register are:

Purpose This register enables the component to switch from functional mode, that is the default behavior, to integration mode where the device inputs and outputs can be directly controlled for the purpose of integration testing.

Note

A device might not operate with the original behavior in integration mode. After performing integration, you must reset the system to ensure the correct behavior of system components that are affected by the integration.

Writing to this register other than when in the disabled state results in UNPREDICTABLE behavior.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes See *Integration register summary* on page 3-7.

Figure 3-12 shows the bit assignments.

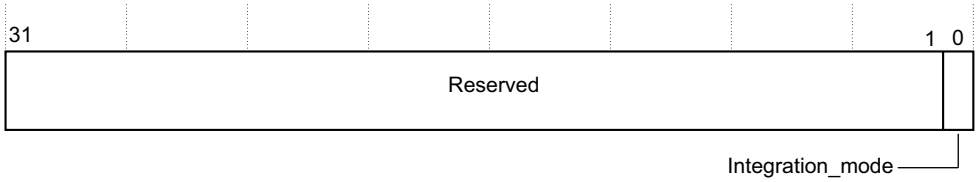


Figure 3-12 ITEN Register bit assignments

Table 3-24 shows the bit assignment.

Table 3-24 ITEN Register bit assignments

Bits	Name	Description
[31:1]	Reserved	Reserved.
[0]	Integration_mode	Enables the component to switch from functional mode to integration mode and back. The possible values for this field are: `0` Disable integration mode. `1` Enable integration mode.

3.6.2 Integration Test Input Register

The characteristics of the ITIP Register are:

Purpose Enables the MMU-400 to read the status of the **spniden** signal.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes See *Integration register summary* on page 3-7.

Figure 3-13 shows the bit assignments.

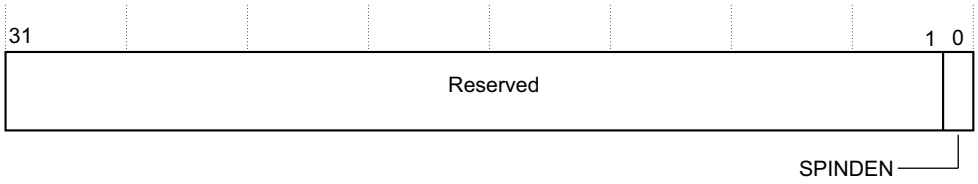


Figure 3-13 ITIP Register bit assignments

Table 3-25 shows the bit assignment.

Table 3-25 ITIP Register bit assignments

Bits	Name	Description
[31:1]	Reserved	Reserved.
[0]	SPNIDEN	The Secure Debug Input, **SPNIDEN**, is reflected in this register at bit 0.

3.6.3 Integration Test Output Register

The characteristics of the ITOP Register are:

Purpose Enables the status of the MMU-400 to be set in integration test mode. In integration mode, this register controls the outputs as Table 3-26 on page 3-35 shows. It can be set in integration mode, by programming the integration control register as Table 3-24 on page 3-33 shows.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes See *Integration register summary* on page 3-7.

Figure 3-14 on page 3-35 shows the bit assignments.

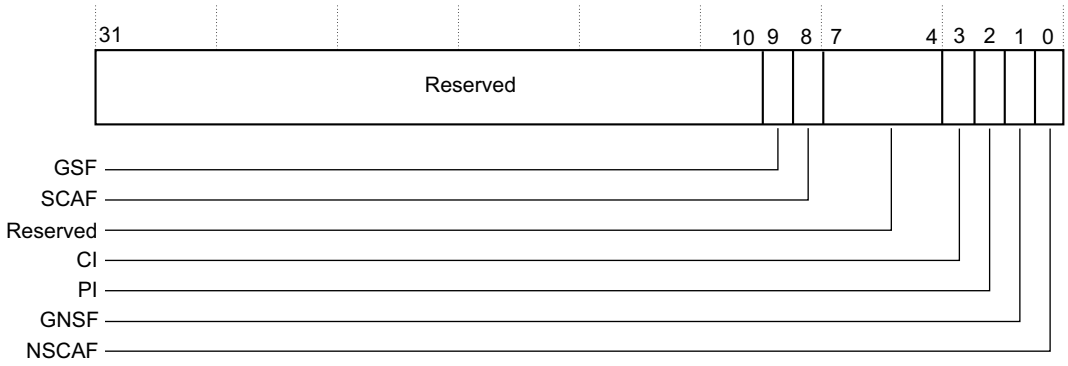


Figure 3-14 ITOP Register bit assignments

Table 3-26 shows the bit assignments

Table 3-26 ITOP Register bit assignments

Bits	Name	Description
[31:10]	-	RESERVED.
[9]	GSF	Global Secure Fault.
[8]	SCAF	Secure Configuration Access Fault.
[7:4]	-	Reserved.
[3]	CI	Context Interrupt.
[2]	PI	Performance Interrupt.
[1]	GNSF	Global Non-secure Fault.
[0]	NSCAF	Non-secure Configuration Access Fault.

3.7 Performance Monitoring registers

This section describes the performance monitoring registers for the MMU-400. It contains the following sections:

- *Performance Monitor Event Count Register*.
- *Performance Monitor Counter Group Configuration Register* on page 3-37.
- *Performance Monitor Counter Group Stream Match Register* on page 3-38.
- *Performance Monitor Configuration Register* on page 3-39.
- *Performance Monitor Control Register* on page 3-40.
- *Performance Monitor Authentication Status register* on page 3-41.
- *Performance Monitor Device Type Register* on page 3-43.

3.7.1 Performance Monitor Event Count Register

The characteristics of the Performance Monitor Event Count Register are:

Purpose The Performance Monitor Event Count register, SMMU_PMEVCNT*n*, is used to read or write the value of the event counter EVCNTR*n*.

The size of the register, SIZE, is defined by the PPMCFGR.SIZE field. See *Performance Monitor Configuration Register* on page 3-39.

Reads from PMEVCNTR*n* return the complete counter value and writes update the complete counter value.

Note

If PMCR.NA==0, you can write to PMEVCNTR*n* even when the counter is disabled. This is true because of one of the following:

- One has been written to the appropriate bit in the PMCTENCLR.
- The PMCR.CEN bit is set to zero.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes *Performance monitoring registers summary* on page 3-7.

Figure 3-15 shows the bit assignments.

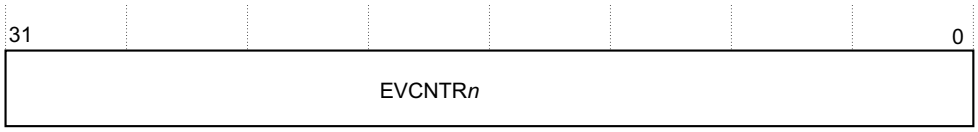


Figure 3-15 Performance Monitoring Event Count Register

Table 3-27 shows the bit assignments.

Table 3-27 Performance Monitoring Event Count Register

Bits	Name	Description
[31:0]	EVCNTR*n*	Value of event counter.

3.7.2 Performance Monitor Counter Group Configuration Register

The characteristics of the Performance Monitor Counter Group Configuration Register are:

Purpose The Performance Monitor Counter Group Configuration register, PMCGCR*n*, controls the behavior of a counter group.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes *Performance monitoring registers summary* on page 3-7.

Figure 3-16 shows the bit assignments.

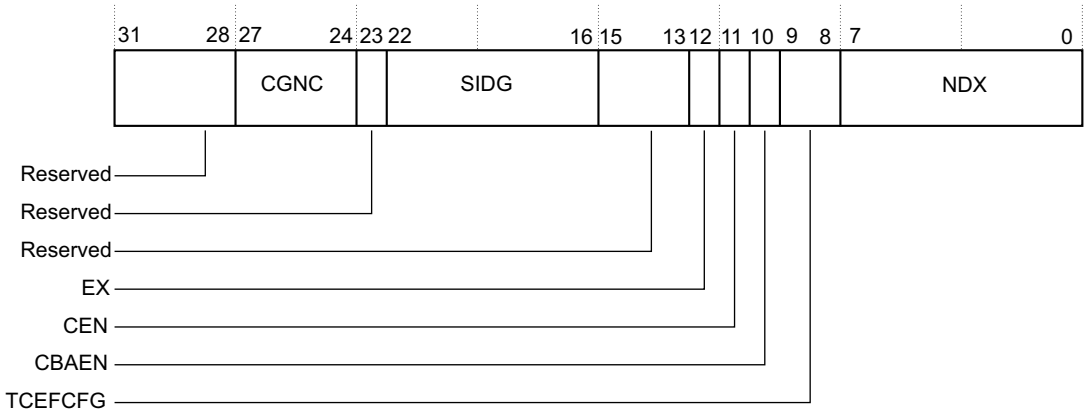


Figure 3-16 Performance Monitor Counter Group Configuration Register bit assignments

Table 3-28 shows the bit assignments.

Table 3-28 Performance Monitor Counter Group Configuration Register bit assignments

Bits	Name	Description
[31:28]	-	Reserved.
[27:24]	CGNC	Counter Group Number of Counters. Indicates the number of counters in this counter group. This field is RO/WI. For the MMU-400, this is fixed at three.
[23]	-	Reserved.
[22:16]	SIDG	StreamID Group. Indicates the StreamID Group that this counter group is affiliated with. This field is RO/WI. For the MMU-400, this is set to 0 to indicate only one group is present.
[15:13]	-	Reserved.
[12]	EX	Export. Corresponds to the Performance Monitor Event Export, PMCR.EX, for this counter group.
[11]	CEN	Count Enable. Corresponds to the Performance Monitor Count Enable, PMCR.CEN, for this counter group.

Table 3-28 Performance Monitor Counter Group Configuration Register bit assignments (continued)

Bits	Name	Description
[10]	CBAEN	Context Bank Assignment Enable. 0 Do not reveal counter group n in translation context bank specified by NDX. 1 Reveal counter group n in translation context bank specified by NDX. If CBAEN==1 and TCEFCFG!=0b10 or 0b01, then UNPREDICTABLE.
[9:8]	TCEFCFG	Translation Context Event Filtering Configuration. 0b00 Count Events on a global basis. 0b01 Count Events restricted to match in corresponding PMCGSMR*n*. 0b10 Count Events restricted to translation context bank indicated by NDX. 0b11 Reserved.
[7:0]	NDX	Index. Interpreted based on TCEFCFG, only valid if TCEFCFG==0b10 else reserved.

3.7.3 Performance Monitor Counter Group Stream Match Register

The characteristics of the Performance Monitor Counter Group Stream Match Register are:

Purpose The Performance Monitor Counter Group Stream Match register, PMCGSMR*n*, specifies the StreamID-based filtering of events counted in a counter group.

The number of ID and MASK bits actually present is configured for Stream_ID_Width. The unimplemented bits behave as RAZ/WI. An implementation provides the same number of ID and MASK bits for every implemented PMCGSMR*n*.

You can enable the StreamID event filtering using the corresponding PMCGCRn.TCEFCFG field.

Configuration The width of the ID and MASK fields is equal to the configured STREAM_ID_WIDTH.

Usage constraints There are no usage constraints.

Attributes *Performance monitoring registers summary* on page 3-7.

Figure 3-17 shows the bit assignments.

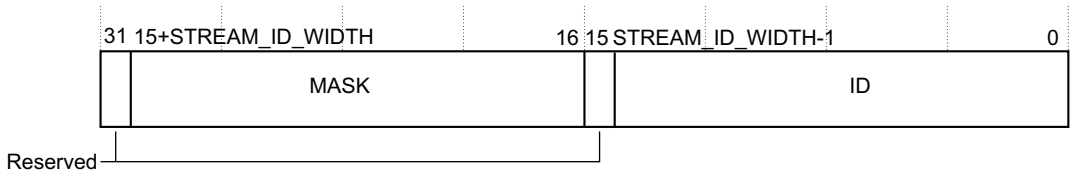


Figure 3-17 Performance Monitor Counter Group Stream Match Register bit assignments

Table 3-29 shows the Performance Monitor Counter Group Stream Match Register bit assignments.

Table 3-29 Performance Monitor Counter Group Stream Match Register bit assignments

Bits	Name	Description
[31]	-	Reserved.
[15+STREAM_ID_WIDTH:16]	MASK	Mask. Identifies if the StreamID is ignored: **MASK[i]**==1 ID[i] is ignored. **MASK[i]**==0 ID[i] is relevant to match.
[15:STREAM_ID_WIDTH]	-	Reserved.
[STREAM_ID_WIDTH-1:0]	ID	StreamID to match.

3.7.4 Performance Monitor Configuration Register

The characteristics of the Performance Monitor Configuration Register are:

Purpose The Performance Monitor Configuration Register, PMCFG, contains *Performance monitoring unit* (PMU)-specific configuration data.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes *Performance monitoring registers summary* on page 3-7

Figure 3-18 shows the bit assignments.

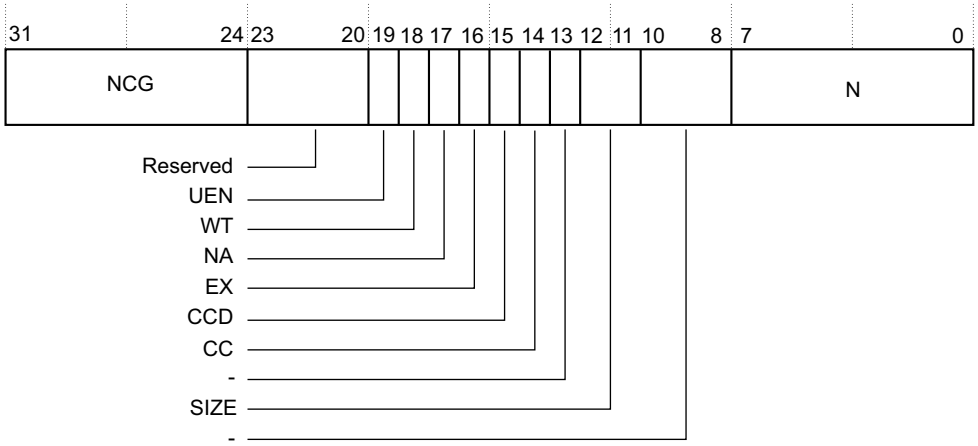


Figure 3-18 Performance Monitor Configuration Register bit assignments

Table 3-30 shows the bit assignments.

Table 3-30 Performance Monitor Configuration Register bit assignments

Bits	Name	Reset value	Description
[31:24]	NCG	0b001	Number of counter groups.
[23:20]	-	-	Reserved.
[19]	UEN	0b000	User Enable. Read as 0, user enable is not supported.

Table 3-30 Performance Monitor Configuration Register bit assignments (continued)

Bits	Name	Reset value	Description
[18]	WT	0b000	Read as 0, two state control state machine is implemented.
[17]	NA	0b000	Reads as 0, you can read the event counters at any time.
[16]	EX	0b001	Event Export. Reads as 1, event export is supported. PMCR.EX is writable.
[15]	CCD	0b000	Cycle Counter pre-scale. Reads as 0, no cycle counter pre-scale.
[14]	CC	0b000	Cycle Counter. Reads as 0, cycle counter not implemented
[13]	Reserved	0b000	Reads as 0.
[12:11]	SIZE	0b011	Counter Size. Reads as 0b011, 32 bit event counters.
[10:8]	Reserved	0b111	Reads as 0b111.
[7:0]	N	0b011	Indicates the number of implemented event counters.

3.7.5 Performance Monitor Control Register

The characteristics of the Performance Monitor Control Register are:

Purpose The Performance Monitor Control Register, PMCR, provides details of the performance monitor implementation, including the number of counters implemented, and configures and controls the counters.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes *Performance monitoring registers summary* on page 3-7.

Figure 3-19 shows the bit assignments.

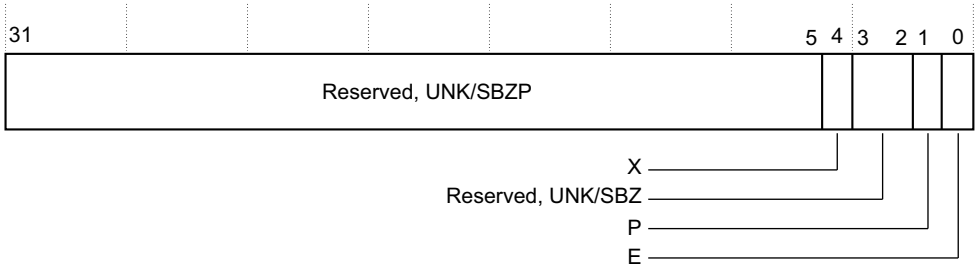


Figure 3-19 Performance Monitor Control Register format

Table 3-31 shows the bit assignments.

Table 3-31 Performance Monitor Control Register format

Bits	Name	Reset value	Description
[31:5]	Reserved, UNK/SBZP	-	Reserved.
[4]	X	0	Export enable. The possible values of this bit are: `0` Export of events is disabled. `1` Export of events is enabled. This bit is used to permit events to be exported to another debug device, such as *Embedded Trace Macrocell* (ETM), over an event bus. If the implementation does not include such an event bus, this bit reads as 0 and ignores writes. This bit does not affect the generation of performance monitor interrupts that can be implemented as a signal exported from the core to an interrupt controller.
[3:2]	Reserved, UNK/SBZP	-	Reserved.
[1]	P	0	Event Counter Reset. This is a WO bit. The effects of writing to this bit are: `0` No action. `1` Reset all event counters to 0. If the cycle counter is implemented, the cycle counter is not reset. **Note** Resetting the event counter does not clear any overflow flags to 0. This bit always reads as 0.
[0]	E	0	Enable. The possible values of this bit are: `0` All counters, including PMCCNTR, are disabled. `1` All counters are enabled. Overflow interrupts are only enabled if the event counters are enabled. Writes to the bit request a stage change. See Table 3-32.

Table 3-32 shows the action on writes to the count enable bit

Table 3-32 Action on writes to the count enable bit

Old value	New value	Action on write
0	0	No action
0	1	Start event
1	0	End event
1	1	No action

3.7.6 Performance Monitor Authentication Status register

The characteristics of the Performance Monitor Authentication Status register are:

Purpose The Performance Monitor Authentication Status registers, SMMU_CBn_AUTHSTATUS, indicates the implemented debug features and provides the current values of the configuration inputs that determine

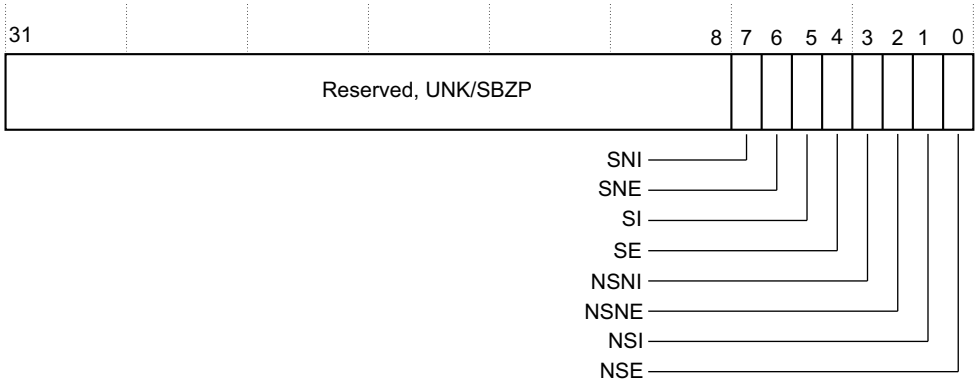
the debug permission. The value returned depends on whether the performance monitor implements the ARM security extension authentication model.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes *Performance monitoring registers summary* on page 3-7.

Figure 3-20 shows the bit assignments.



Note: If Security Extensions are not implemented, then bits [3:2] are assigned to NSNI and NSNE.

Figure 3-20 Performance Monitor Authentication Status Register format

Table 3-33 shows the bit assignments.

Table 3-33 Performance Monitor Authentication Status Register format

Bits	Name	Reset value	Description
[31:8]	Reserved, UNK/SBZP	-	Reserved.
[7]	SNI	0	This bit is RAZ, because Secure non-invasive debug features are not implemented.
[6]	SNE	0	This bit is RAZ, because Secure non-invasive debug features cannot be enabled.
[5]	SI	1	Secure invasive debug features implemented. This bit is RAO, because Secure invasive debug features are implemented.
[4]	SE	1	Secure invasive debug is not enabled. This bit is RAO.
[3]	NSNI	0	This bit is 0, as Non-secure non-invasive debug is not supported.
[2]	NSNE	0	This bit is RAZ, as Non-secure non invasive debug cannot be enabled.
[1]	NSI	0	This bit is RAZ, Non-secure invasive debug features are not implemented.
[0]	NSE	0	Non-secure invasive debug is not enabled. This bit is RAZ.

3.7.7 Performance Monitor Device Type Register

The characteristics of the Performance Monitor Device Type register are:

Purpose The Performance Monitor Device Type register, PMDEVTYPE, provides the CoreSight device type information for the performance monitors, and indicates the type of debug component. You must implement the PMDEVTYPE register in all CoreSight components.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes *Performance monitoring registers summary* on page 3-7

Figure 3-21 shows the bit assignments.

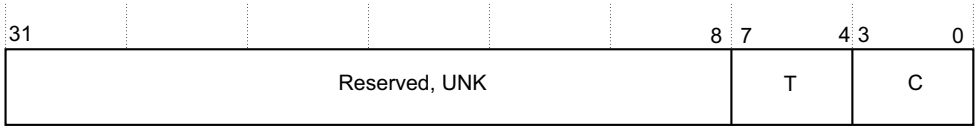


Figure 3-21 Performance Monitor Device Type Register bit assignments

Table 3-34 shows the bit assignments.

Table 3-34 Performance Monitor Device Type Register bit assignments

Bits	Name	Reset value	Description
[31:8]	Reserved, UNK	-	Reserved.
[7:4]	T	`0x5`	Sub-type, a fixed value of `0x5` which indicates association with a memory management unit conforming to the *ARM® System Memory Management Unit Architecture*.
[3:0]	C	`0x6`	Class, a fixed value of `0x6`, which indicates a Performance Monitor Device Type.

3.8 The MMU-400 Security State Determination Address Space

The MMU-400 security state determination address space characteristics are:

Purpose The MMU-400 security state determination address space, SMMU_SSDR, is part of the translation process.

The security state determination address space provides an indication of whether each SSD_Index is acting for the Secure or Non-secure domains. The address space is only accessible to Secure memory transactions.

One bit is provided for each SSD_Index value. The MMU-400 supports an SSD_Index of up to 15 bits in size, corresponding to a total possible indication state of 4KB.

The address space might not be fully populated, depending on the implemented PAGESIZE and the SSD_Index width. The SSD_Index width can be read from SMMU_IDR1.NUMSSDNDXB. Unimplemented SSD_Index bits behave as though they are zero. The Security state determination register bits corresponding to values above the implemented SSD_Index size behave as UNK/SBOP.

The security state determination bits can have fixed values that correspond to SSD_Index values having a fixed Secure or Non-secure ownership. Software can detect programmable bits by using a read-modify-write sequence.

The security state determination registers are implemented as *Configurable options* on page 1-6 describes. The MMU-400 implementation and the system it is integrated in can use alternative means to resolve the security status of transactions. SMMU_IDR1.SSDTP indicates the presence of the security state determination table. In an implementation where the registers are not present, this address space behaves as UNK/SBOP.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes *The MMU-400 security state determination address space summary* on page 3-8.

Table 3-35 shows the security state determination address space layout in terms of offsets from SMMU_GSSD_BASE.

Table 3-35 Security state determination address space

Offset	Name	Description
0x000	SMMU_SSDR0	Corresponding to SSD_Index values 0-31
0x004	SMMU_SSDR1	Corresponding to SSD_Index values 32-63
0x008 - 0xFFC	SMMU_SSDR1023 - SMMU_SSDR2	Corresponding to SSD_Index values 64-32767
0x01000 - (*PAGESIZE* − 0x4)	Reserved	Reserved

If the security state determination register space is implemented, the behavior of each SMMU_SSDRn bit is:

```
// SMMU_SSDRn selected using SSD_Index<15:5>
if (SMMU_SSDRn[SSD_Index<4:0>] == 1) {
    // Transaction is Non-secure
```

```
} else {
    // Transaction is Secure
}
```

3.9 Peripheral and Component Identification registers

This section describes the following identification registers:

- *Component Identification registers*.
- *Peripheral Identification registers*.

3.9.1 Component Identification registers

The characteristics of the Component Identification registers, CID are:

Purpose Bits[7:0] of the CID 0-3 registers hold preamble information and bits[31:8] are reserved.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes See *Peripheral and Component identification registers summary* on page 3-8.

Figure 3-22 shows the bit assignments.

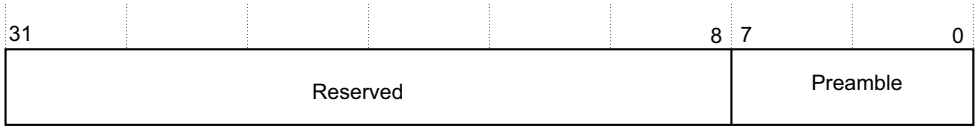


Figure 3-22 CID Registers 0-3 bit assignments

Table 3-36 shows the bit assignments

Table 3-36 CID Register 0-3 bit assignments

CID	Bits	Name	Reset value	Description
0	[7:0]	Preamble	0x0D	Preamble
1	[7:0]	Preamble	0xF0	Preamble
2	[7:0]	Preamble	0x05	Preamble
3	[7:0]	Preamble	0xB1	Preamble

3.9.2 Peripheral Identification registers

The characteristics of the Peripheral Identification registers, Peripheral ID registers are:

Purpose Only bits[7:0] of each register are used. The peripheral ID registers 7-5 are reserved.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attributes See *Peripheral and Component identification registers summary* on page 3-8.

The following are the Peripheral Identification registers:

- *Peripheral Identification Register 0* on page 3-47.
- *Peripheral Identification Register 1* on page 3-47.

- *Peripheral Identification Register 2*.
- *Peripheral Identification Register 3* on page 3-48.
- *Peripheral Identification Register 4* on page 3-48.
- *Peripheral Identification registers 5-7* on page 3-49.

Peripheral Identification Register 0

Figure 3-23 shows the bit assignments.

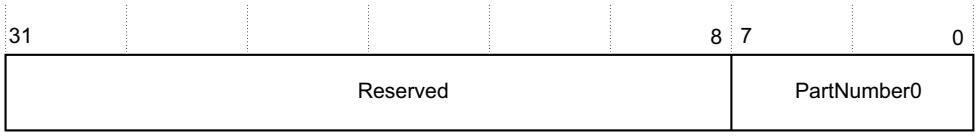


Figure 3-23 PID Register 0 register bit assignments

Table 3-37 shows the bit assignments.

Table 3-37 PID Register 0 register bit assignments

Bits	Name	Reset value	Description
[31:8]	Reserved	-	Reserved.
[7:0]	PartNumber0	0x80	Middle and lower-packed BCD value of device number [7:0].

Peripheral Identification Register 1

Figure 3-24 shows the bit assignments.

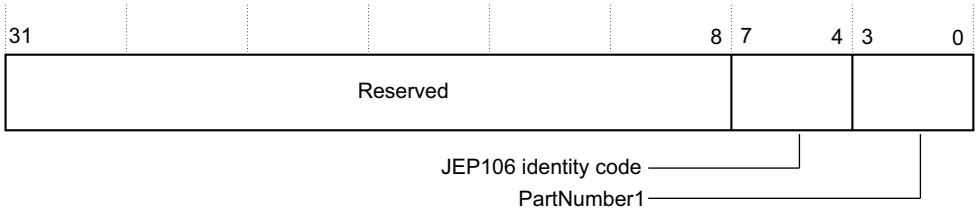


Figure 3-24 PID Register 1 register bit assignments

Table 3-38 shows the bit assignments.

Table 3-38 PID Register 1 register bit assignments

Bits	Name	Reset value	Description
[31:8]	Reserved	-	Reserved.
[7:4]	JEP106 identity code	0xB	JEP106 identity code.
[3:0]	PartNumber1	0x4	Upper packed-BCD value of device number [11:8].

Peripheral Identification Register 2

Figure 3-25 on page 3-48 shows the bit assignments.

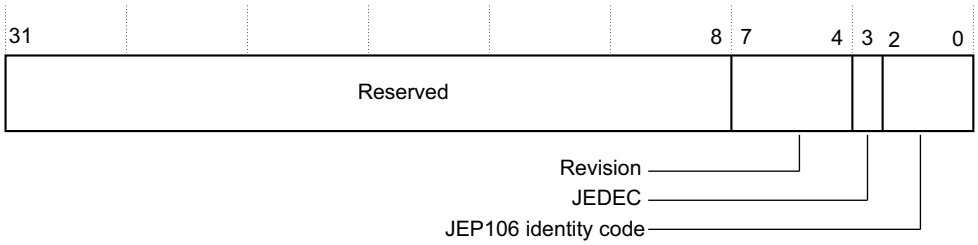


Figure 3-25 PID Register 2 register bit assignments

Table 3-39 shows the bit assignments

Table 3-39 PID Register 2 register bit assignments

Bits	Name	Reset value	Description
[31:8]	Reserved	-	Reserved.
[7:4]	Revision	0x1	Revision number of peripheral. A value of 1 indicates r0p1.
[3]	JEDEC	1	Always set, indicates that a JEDEC assigned value is used.
[2:0]	JEP106 identity code	b011	JEP106 identity code [6:4].

Peripheral Identification Register 3

Figure 3-26 shows the bit assignments.

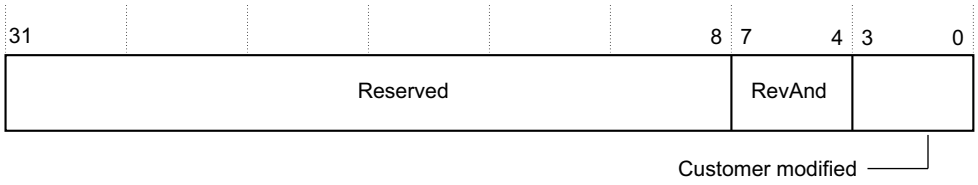


Figure 3-26 PID Register 3 register bit assignments

Table 3-40 shows the bit assignments

Table 3-40 PID Register 3 register bit assignments

Bits	Name	Reset value	Description
[31:8]	Reserved	-	Reserved.
[7:4]	RevAnd	0	RevAnds at top-level.
[3:0]	Customer Modified	0	Customer modified number. It must be 0x0 from ARM.

Peripheral Identification Register 4

Figure 3-27 on page 3-49 shows the bit assignments.

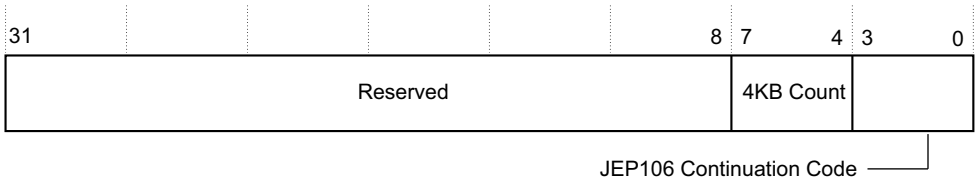


Figure 3-27 PID Register 4 register bit assignments

Table 3-41 shows the bit assignments

Table 3-41 PID Register 4 register bit assignments

Bits	Name	Reset value	Description
[31:8]	Reserved	-	Reserved.
[7:4]	4KB Count.	0x4	4KB Count.
[3:0]	JEP106 continuation code.	0x4	JEP106 continuation code.

Peripheral Identification registers 5-7

Figure 3-28 shows the bit assignments.

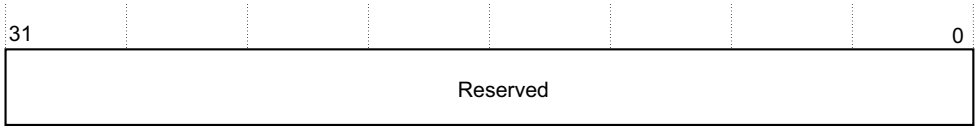


Figure 3-28 PID register 5-7 bit assignments

Table 3-42 shows the bit assignments

Table 3-42 PID register 5-7 bit assignments

Bits	Name	Reset value	Description
[31:0]	Reserved	0	Reserved

3.10 Translation Context Bank registers

This section describes the translation context bank registers.

3.10.1 System Control Register

The characteristics of the System Control Register are:

Purpose The System Control Register, SMMU_CB*n*_SCTLR, provides top-level control of the translation system for the related context bank.

Configuration Available in all MMU-400 configurations.

Usage constraints There are no usage constraints.

Attribute See *Translation context bank registers summary* on page 3-8.

Figure 3-29 shows the bit assignment.

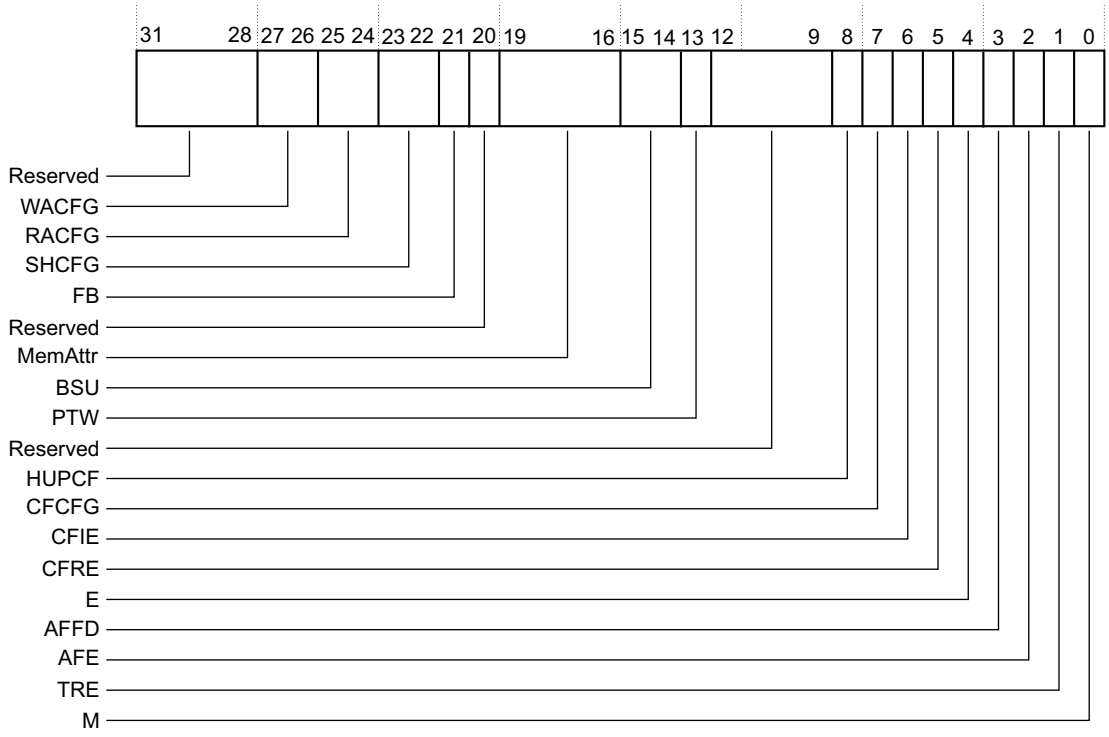


Figure 3-29 System Control Register bit assignment

Table 3-43 shows the bit assignment.

Table 3-43 System Control Register bit assignment

Bits	Name	Reset value	Description
[31:28]	Reserved	-	Reserved.
[27:26]	WACFG	-	Write Allocate Configuration. The encodings of this field are: `0b00` Use the default allocation attributes. `0b01` Reserved. `0b10` Write-Allocate. `0b11` No Write-Allocate. **Note** This field applies to the processing of transactions where the context bank translation is disabled, that is, where SMMU_CB*n*_SCTLR.M has the value 0.
[25:24]	RACFG	-	Read Allocate Configuration. Controls the allocation hint for read transactions where the context bank is disabled. The encodings of this field are: `0b00` Use the default allocation attributes. `0b01` Reserved. `0b10` Read-Allocate. `0b11` No Read-Allocate. **Note** This field applies to the processing of transactions where the context bank translation is disabled, that is, where SMMU_CB*n*_SCTLR.M has the value 0.
[23:22]	SHCFG	-	Shared Configuration. Controls the shareable attributes for transactions where the context bank is disabled. The encodings of this field are: `0b00` Use shareable attribute as presented with transaction. `0b01` Outer shareable. `0b10` Inner shareable. `0b11` Non-shareable. **Note** This field applies to the processing of transactions where the context bank translation is disabled, that is, where SMMU_CB*n*_SCTLR.M has the value 0.
[21]	FB	-	Force Broadcast. This field forces the broadcast of TLB maintenance, BPIALL and ICIALLU operations.
[20]	Reserved	-	Reserved.
[19:16]	MemAttr	-	Memory Attribute. The memory attributes can be overlaid if SMMU_CB*n*_SCTLR.M has the value 0. Table 3-44 on page 3-53 and Table 3-45 on page 3-53 show valid values for this field.

Table 3-43 System Control Register bit assignment (continued)

Bits	Name	Reset value	Description
[15:14]	BSU	-	Barrier Shareability Upgrade. This field upgrades the required shareability domain of barriers issued by client devices mapped to this stream mapping register group by setting the minimum shareability domain that is applied to any barrier. The encodings of this field are: `0b00` No effect. `0b01` Inner shareable. `0b10` Outer shareable. `0b11` Full system.
[13]	Reserved	-	Reserved.
[12:9]	Reserved	-	Reserved.
[8]	HUPCF	-	Hit Under Previous Context Fault. The possible values of this Hit-under-fault bit are: `0` Stall or terminate subsequent transactions in the presence of an outstanding context fault. `1` Process subsequent transactions independent of an outstanding context fault.
[7]	CFCFG	-	Context Fault Configuration. The possible value of this bit is: `0` Terminate.
[6]	CFIE	0	Context Fault Interrupt Enable. The possible values of this bit are: `0` Do not raise an interrupt when a Context fault occurs. `1` Raise an interrupt when a Context fault occurs. This field resets to the value 0.
[5]	CFRE	0	Context Fault Report Enable. The possible values of this bit are: `0` Do not return an abort when a Context fault occurs. `1` Return an abort when a Context fault occurs.
[4]	E	-	Endianess. This field indicates the endianess format of translation table entries. The possible values of this bit are: `0` Little Endian format. `1` Big Endian format.
[3]	AFFD	-	Access Flag Fault Disable. This field determines whether access flag faults are reported if they are raised. The possible values of this bit are: `0` Access flag faults are reported. `1` Access flag faults are not reported.
[2]	AFE	1	Access Flag Enable. This bit is UNK/SBOP.
[1]	TRE	1	TEX Remap Enable. This bit is UNK/SBOP.
[0]	M	0	MMU Enable. This is a global enable bit for the involved translation context bank. The possible values of this bit are: `0` MMU behavior for this translation context bank is disabled. `1` MMU behavior for this translation context bank is enabled.

Table 3-44 shows MemAttr bit values

Table 3-44 MemAttr bit values

Bits[3:2]	Meaning
0b00	Strongly-ordered or device memory
0b01	Outer non-cacheable normal memory
0b10	Outer write-through normal memory
0b11	Outer write-back normal memory

Table 3-45 shows secondary MemAttr bit values

Table 3-45 Secondary MemAttr bit values

Bits[1:0]	Meaning when bits[3:2] == 0b00	Meaning when bits[3:2]!= 0b00
0b00	Strongly-ordered	Reserved
0b01	Device	Inner non-cacheable normal memory
0b10	Reserved	Inner write-through normal memory
0b11	Reserved	Inner write-back normal memory

3.10.2 Translation Table Base Control Register

The characteristics of the Translation Table Base Control Register are:

Purpose SMMU_CBn_TTBCR, the Translation Table Base Control Register, provides additional configuration for the translation process.

Configuration Available in all MMU-400 configurations.

Usage constraints Bit[31] is RAO/WI.

Attributes See *Translation context bank registers summary* on page 3-8.

Figure 3-30 shows the bit assignments.

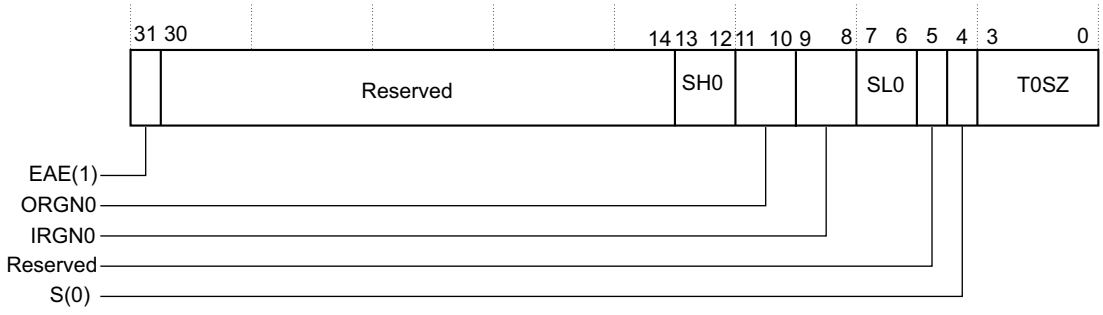


Figure 3-30 Translation Table Control Register bit assignments

Table 3-46 shows the bit assignments.

Table 3-46 Translation Table Control Register bit assignments

Bits	Name	Description
[31]	EAE (1)	Extended Address Enable. This field always reads as the value 1. Writes are ignored. A value of 1 means use the translation system defined in the LPAE.
[30:14]	Reserved	Reserved.
[13:12]	SH0	Shareability attributes for the memory associated with the translation table walks using TTBR0.
[11:10]	ORGN0	Outer cacheability attributes for the memory associated with the translation table walks using SMMU_CB*n*_TTBR0.
[9:8]	IRGN0	Inner cacheability attributes for the memory associated with the translation table walks using SMMU_CB*n*_TTBR0.
[7]	Reserved	Reserved.
[6]	SL0	When bit [6] is 0, then the starting Level for SMMU_CB*n*_TTBR0 addressed region is Level 2. When bit [6] is 1, then the starting Level for SMMU_CB*n*_TTBR0 addressed region is Level 1.
[5]	Reserved	Reserved.
[4]	S(0)	This bit must be programmed to T0SZ[3], or the effect is UNPREDICTABLE. This bit is a sign extension of the T0SZ field, and is allocated this way for future compatibility for translation table systems with a larger input address.
[3:0]	T0SZ	The Size offset of the SMMU_CB*n*_TTBR0 addressed region, encoded as a 4 bits signed number giving the size of the region as $2^{32-T0SZ}$.

Appendix A
Signal Descriptions

This appendix describes the MMU-400 signals in the following sections:

A.1 Clock and resets

This section describes the clock and reset signals of the MMU-400.

Table A-1 shows the clock and reset signals for the PTW block.

Table A-1 PTW block clock and reset signals

Signal	Width	Direction	Description
cclk	1	Input	Clock for the PTW block
cresetn	1	Input	Reset for the PTW block

Table A-2 shows the clock and reset signals for the TLB block.

Table A-2 TLB block clock and reset signals

Signal	Width	Direction	Description
bclk	1	Input	Clock for the TLB block
bresetn	1	Input	Reset for the TLB block

A.2 AMBA signals

The *ARM® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite* describes the AMBA AXI signals, ACE-Lite signals, and AXI LPI signals that the MMU-400 uses. This section describes:

- *AXI3 signals*.
- *AXI4 signals* on page A-7.
- *ACE-Lite signals* on page A-11.
- *APB signals* on page A-16.
- *AXI low-power interface signals* on page A-17.
- *Snoop channel signals* on page A-17.

A.2.1 AXI3 signals

The following sections describe the AXI3 signals:

- *Write address channel signals*.
- *Write data channel signals* on page A-4.
- *Write response channel signals* on page A-5,
- *Read address channel signals* on page A-5,
- *Read data channel signals* on page A-6.

See the *ARM® CoreLink™ MMU-400 System Memory Management Unit AMBA® Designer (ADR-400) User Guide Supplement* for more information on the following ID widths:

- Master ID width, I_M.
- Slave ID width, I_S.
- PTW ID width, I_P.

Write address channel signals

Table A-3 shows the AXI3 write address channel signals.

Table A-3 Write address channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master Port of TLB block	Direction	Master Port of PTW block	Direction
AWID	**awid_s[I_S:0]**	Input	**awid_m[I_M:0]**	Output	**awid_ptw[I_P:0]**	Output
AWADDR	**awaddr_s[39:0]**	Input	**awaddr_m[39:0]**	Output	**awaddr_ptw[39:0]**	Output
AWLEN	**awlen_s[3:0]**	Input	**awlen_m[3:0]**	Output	**awlen_ptw[3:0]**	Output
AWSIZE	**awsize_s[2:0]**	Input	**awsize_m[2:0]**	Output	**awsize_ptw[2:0]**	Output
AWBURST	**awburst_s[1:0]**	Input	**awburst_m[1:0]**	Output	**awburst_ptw[1:0]**	Output
AWLOCK	**awlock_s[1:0]**	Input	**awlock_m[1:0]**	Output	**awlock_ptw[1:0]**	Output
AWCACHE	**awcache_s[3:0]**	Input	**awcache_m[3:0]**	Output	**awcache_ptw[3:0]**	Output
AWPROT	**awprot_s[2:0]**	Input	**awprot_m[2:0]**	Output	**awprot_ptw[2:0]**	Output
AWVALID	**awvalid_s[0]**	Input	**awvalid_m[0]**	Output	**awvalid_ptw[0]**	Output
AWUSER	**awuser_s[AWUSER_WIDTH-1:0]**[a]	Input	**awuser_m[AWUSER_WIDTH+5:0]**	Output	-	-
AWREADY	**awready_s[0]**	Output	**awready_m[0]**	Input	**awready_ptw[0]**	Input

a. AWUSER_WIDTH is the width of AXI slave interface **AWUSER** signal.

Note

- The PTW signals are present only when a separate AXI configuration option is selected.
- The write address, write data, and write response signals of the PTW block are dummy signals that are unused.

Write data channel signals

Table A-4 shows the AXI3 write data channel signals for the slave port of the TLB block.

Table A-4 Write data channel signals-slave port of TLB block

AMBA equivalent	Slave port of TLB block		Direction
WID	**wid_s[I_S:0]**		Input
WDATA	**For 64-bit** **For 128-bit**	The data width is **wdata_s[63:0]** The data width is **wdata_s[127:0]**	Input
WSTRB	**For 64-bit** **For 128-bit**	The data width is **wstrb_s[7:0]** The data width is **wstrb_s[15:0]**	Input
WLAST	**wlast_s[0]**		Input
WVALID	**wvalid_s[0]**		Input
WUSER	**wuser_s[WUSER_WIDTH-1:0]**[a]		Input
WREADY	**wready_s[0]**		Output

a. WUSER_WIDTH is the width of AXI slave interface **WUSER** signal.

Table A-5 shows the AXI3 write data channel signals for the master port of the TLB block.

Table A-5 Write data channel signals-master port of TLB block

AMBA equivalent	Master Port of TLB block		Direction
WID	**wid_m[I_M:0]**		Output
WDATA	**For 64-bit** **For 128-bit**	The data width is **wdata_m[63:0]** The data width is **wdata_m[127:0]**	Output
WSTRB	**For 64-bit** **For 128-bit**	The data width is **wstrb_m[7:0]** The data width is **wstrb_m[15:0]**	Output
WLAST	**wlast_m[0]**		Output
WVALID	**wvalid_m[0]**		Output
WUSER	**wuser_m[WUSER_WIDTH-1:0]**		Output
WREADY	**wready_m[0]**		Input

Table A-6 shows the AXI3 write data channel signals for the master port of the PTW block.

Table A-6 Write data channel signals-master port of PTW block

AMBA equivalent	Master Port of PTW block		Direction
WID	**wid_ptw[I_P:0]**		Output
WDATA	**For 64-bit** **For 128-bit**	The data width is **wdata_ptw[63:0]** The data width is **wdata_ptw[127:0]**	Output
WSTRB	**For 64-bit** **For 128-bit**	The data width is **wstrb_ptw[7:0]** The data width is **wstrb_ptw[15:0]**	Output
WLAST	**wlast_ptw[0]**		Output
WVALID	**wvalid_ptw[0]**		Output
WUSER	-		-
WREADY	**wready_ptw[0]**		Input

Write response channel signals

Table A-7 shows the AXI3 write response channel signals.

Table A-7 Write response channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW block	Direction
BID	**bid_s[I_S:0]**	Output	**bid_m[I_M:0]**	Input	**bid_ptw[I_P:0]**	Input
BRESP	**bresp_s[1:0]**	Output	**bresp_m[1:0]**	Input	**bresp_ptw[1:0]**	Input
BVALID	**bvalid_s[0]**	Output	**bvalid_m[0]**	Input	**bvalid_ptw[0]**	Input
BUSER	**buser_s[BUSER_WIDTH-1:0]**[a]	Output	**buser_m[BUSER_WIDTH-1:0]**	Input	-	-
BREADY	**bready_s[0]**	Input	**bready_m[0]**	Output	**bready_ptw[0]**	Output

a. BUSER_WIDTH is the width of AXI slave interface **BUSER** signal.

Read address channel signals

Table A-8 shows the AXI3 read address channel signals.

Table A-8 Read address channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW block	Direction
ARID	**arid_s[I_S:0]**	Input	**arid_m[I_M:0]**	Output	**arid_ptw[I_P:0]**	Output
ARADDR	**araddr_s[39:0]**	Input	**araddr_m[39:0]**	Output	**araddr_ptw[39:0]**	Output
ARLEN	**arlen_s[3:0]**	Input	**arlen_m[3:0]**	Output	**arlen_ptw[3:0]**	Output
ARSIZE	**arsize_s[2:0]**	Input	**arsize_m[2:0]**	Output	**arsize_ptw[2:0]**	Output
ARBURST	**arburst_s[1:0]**	Input	**arburst_m[1:0]**	Output	**arburst_ptw[1:0]**	Output

Table A-8 Read address channel signals (continued)

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW block	Direction
ARLOCK	**arlock_s[1:0]**	Input	**arlock_m[1:0]**	Output	**arlock_ptw[1:0]**	Output
ARCACHE	**arcache_s[3:0]**	Input	**arcache_m[3:0]**	Output	**arcache_ptw[3:0]**	Output
ARPROT	**arprot_s[2:0]**	Input	**arprot_m[2:0]**	Output	**arprot_ptw[2:0]**	Output
ARVALID	**arvalid_s[0]**	Input	**arvalid_m[0]**	Output	**arvalid_ptw[0]**	Output
ARUSER	**aruser_s[ARUSER_WIDTH-1:0]**[a]	Input	**aruser_m[ARUSER_WIDTH+5:0]**	Output	**aruser_ptw[5:0]**	Output
ARREADY	**arready_s[0]**	Output	**arready_m[0]**	Input	**arready_ptw[0]**	Input

a. ARUSER_WIDTH is the width of AXI slave interface **ARUSER** signal.

Read data channel signals

Table A-9 shows the AXI3 read data channel signals for the slave port of the TLB block.

Table A-9 Read data channel signals-slave port of TLB block

AMBA equivalent	Slave port of TLB block		Direction
RID	**rid_s[I_S:0]**		Output
RDATA	**For 64-bit** **For 128-bit**	The data width is **rdata_s[63:0]** The data width is **rdata_s[127:0]**	Output
RRESP	**rresp_s[1:0]**		Output
RLAST	**rlast_s[0]**		Output
RVALID	**rvalid_s[0]**		Output
RUSER	**ruser_s[RUSER_WIDTH-1:0]**[a]		Output
RREADY	**rready_s[0]**		Input

a. RUSER_WIDTH is the width of AXI slave interface **RUSER** signal.

Table A-10 shows the AXI3 read data channel signals for the master port of the TLB block.

Table A-10 Read data channel signals-master port of TLB block

AMBA equivalent	Master port of TLB block		Direction
RID	**rid_m[I_M:0]**		Input
RDATA	**For 64-bit** **For 128-bit**	The data width is **rdata_m[63:0]** The data width is **rdata_m[127:0]**	Input
RRESP	**rresp_m[1:0]**		Input
RLAST	**rlast_m[0]**		Input

Table A-10 Read data channel signals-master port of TLB block (continued)

AMBA equivalent	Master port of TLB block	Direction
RVALID	**rvalid_m[0]**	Input
RUSER	**ruser_m[RUSER_WIDTH-1:0]**	Input
RREADY	**rready_m[0]**	Output

Table A-11 shows the AXI3 read data channel signals for the master port of the PTW block.

Table A-11 Read data channel signals-master port of PTW

AMBA equivalent	Master port of PTW block	Direction
RID	**rid_ptw[I_P:0]**	Input
RDATA	**For 64-bit** The data width is **rdata_ptw[63:0]** **For 128-bit** The data width is **rdata_ptw[127:0]**	Input
RRESP	**rresp_ptw[1:0]**	Input
RLAST	**rlast_ptw[0]**	Input
RVALID	**rvalid_ptw[0]**	Input
RUSER	-	-
RREADY	**rready_ptw[0]**	Output

A.2.2 AXI4 signals

The following sections describe the AXI4 signals:

- *Write address channel signals* on page A-8.
- *Write data channel signals* on page A-8.
- *Write response channel signals* on page A-9.
- *Read address channel signals* on page A-10.
- *Read data channel signals* on page A-10.

See the *ARM® CoreLink™ MMU-400 System Memory Management Unit AMBA® Designer (ADR-400) User Guide Supplement* for more information on the following ID widths:

- Master ID width, I_M.
- Slave ID width, I_S.
- PTW ID width, I_P.

Write address channel signals

Table A-12 shows the AXI4 write address channel signals.

Table A-12 Write address channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master Port of PTW block	Direction
AWID	**awid_s[I_S:0]**	Input	**awid_m[I_M:0]**	Output	**awid_ptw[I_P:0]**	Output
AWADDR	**awaddr_s[39:0]**	Input	**awaddr_m[39:0]**	Output	**awaddr_ptw[39:0]**	Output
AWLEN	**awlen_s[7:0]**	Input	**awlen_m[7:0]**	Output	**awlen_ptw[7:0]**	Output
AWSIZE	**awsize_s[2:0]**	Input	**awsize_m[2:0]**	Output	**awsize_ptw[2:0]**	Output
AWBURST	**awburst_s[1:0]**	Input	**awburst_m[1:0]**	Output	**awburst_ptw[1:0]**	Output
AWLOCK	**awlock_s[0]**	Input	**awlock_m[0]**	Output	**awlock_ptw[0]**	Output
AWCACHE	**awcache_s[3:0]**	Input	**awcache_m[3:0]**	Output	**awcache_ptw[3:0]**	Output
AWPROT	**awprot_s[2:0]**	Input	**awprot_m[2:0]**	Output	**awprot_ptw[2:0]**	Output
AWVALID	**awvalid_s[0]**	Input	**awvalid_m[0]**	Output	**awvalid_ptw[0]**	Output
AWREGION	**awregion_s[3:0]**	Input	**awregion_m[3:0]**	Output	**awregion_ptw[3:0]**	Output
AWQOS	**awqos_s[3:0]**	Input	**awqos_m[3:0]**	Output	**awqos_ptw[3:0]**	Input
AWUSER	**awuser_s[AWUSER_WIDTH-1:0]**	Input	**awuser_m[AWUSER_WIDTH+5:0]**	Output	-	-
AWREADY	**awready_s[0]**	Output	**awready_m[0]**	Input	**awready_ptw[0]**	Output

Write data channel signals

Table A-13 shows the AXI4 write data channel signals for the slave port of the TLB block.

Table A-13 Write data channel signals-slave port of TLB block

AMBA equivalent	Slave port of TLB block		Direction
WDATA	**For 64-bit** **For 128-bit**	The data width is **wdata_s[63:0]** The data width is **wdata_s[127:0]**	Input
WSTRB	**For 64-bit** **For 128-bit**	The data width is **wstrb_s[7:0]** The data width is **wstrb_s[15:0]**	Input
WLAST	**wlast_s[0]**		Input
WVALID	**wvalid_s[0]**		Input
WREADY	**wready_s[0]**		Output
WUSER	**wuser_s[WUSER_WIDTH-1:0]**		Input

Table A-14 shows the AXI4 write data channel signals for the master port of the TLB block.

Table A-14 Write data channel signals-master port of TLB block

AMBA equivalent	Master port of TLB block		Direction
WDATA	**For 64-bit** **For 128-bit**	The data width is **wdata_m[63:0]** The data width is **wdata_m[127:0]**	Output
WSTRB	**For 64-bit** **For 128-bit**	The data width is **wstrb_m[7:0]** The data width is **wstrb_m[15:0]**	Output
WLAST	**wlast_m[0]**		Output
WVALID	**wvalid_m[0]**		Output
WREADY	**wready_m[0]**		Input
WUSER	**wuser_m[WUSER_WIDTH-1:0]**		Output

Table A-15 shows the AXI4 write data channel signals for the PTW block.

Table A-15 Write data channel signals-master port of PTW block

AMBA equivalent	Master Port of PTW block		Direction
WDATA	**For 64-bit** **For 128-bit**	The data width is **wdata_ptw[63:0]** The data width is **wdata_ptw[127:0]**	Output
WSTRB	**For 64-bit** **For 128-bit**	The data width is **wstrb_ptw[7:0]** The data width is **wstrb_ptw[15:0]**	Output
WLAST	**wlast_ptw[0]**		Output
WVALID	**wvalid_ptw[0]**		Output
WREADY	**wready_ptw[0]**		Input
WUSER	-		-

Write response channel signals

Table A-16 shows the AXI4 write response channel signals.

Table A-16 Write response channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW block	Direction
BID	**bid_s[I_S:0]**	Output	**bid_m[I_M:0]**	Input	**bid_ptw[I_P:0]**	Input
BRESP	**bresp_s[1:0]**	Output	**bresp_m[1:0]**	Input	**bresp_ptw[1:0]**	Input
BVALID	**bvalid_s[0]**	Output	**bvalid_m[0]**	Input	**bvalid_ptw[0]**	Input
BUSER	**buser_s[BUSER_WIDTH-1:0]**	Output	**buser_m[BUSER_WIDTH-1:0]**	Input	-	-
BREADY	**bready_s[0]**	Input	**bready_m[0]**	Output	**bready_ptw[0]**	Output

Read address channel signals

Table A-17 shows the AXI4 read address channel signals.

Table A-17 Read address channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW block	Direction
ARID	**arid_s[I_S:0]**	Input	**arid_m[I_M:0]**	Output	**arid_ptw[I_P:0]**	Output
ARADDR	**araddr_s[39:0]**	Input	**araddr_m[39:0]**	Output	**araddr_ptw[39:0]**	Output
ARLEN	**arlen_s[7:0]**	Input	**arlen_m[7:0]**	Output	**arlen_ptw[7:0]**	Output
ARSIZE	**arsize_s[2:0]**	Input	**arsize_m[2:0]**	Output	**arsize_ptw[2:0]**	Output
ARBURST	**arburst_s[1:0]**	Input	**arburst_m[1:0]**	Output	**arburst_ptw[1:0]**	Output
ARLOCK	**arlock_s[0]**	Input	**arlock_m[0]**	Output	**arlock_ptw[0]**	Output
ARCACHE	**arcache_s[3:0]**	Input	**arcache_m[3:0]**	Output	**arcache_ptw[3:0]**	Output
ARPROT	**arprot_s[2:0]**	Input	**arprot_m[2:0]**	Output	**arprot_ptw[2:0]**	Output
ARVALID	**arvalid_s[0]**	Input	**arvalid_m[0]**	Output	**arvalid_ptw[0]**	Output
ARREGION	**arregion_s[3:0]**	Input	**arregion_m[3:0]**	Output	**arregion_ptw[3:0]**	Output
ARQOS	**arqos_s[3:0]**	Input	**arqos_m[3:0]**	Output	**arqos_ptw[3:0]**	Output
ARUSER	**aruser_s[ARUSER_WIDTH-1:0]**	Input	**aruser_m[ARUSER_WIDTH+5:0]**	Output	**aruser_ptw[5:0]**	Output
ARREADY	**arready_s[0]**	Output	**arready_m[0]**	Input	**arready_ptw[0]**	Input

Read data channel signals

Table A-18 shows the AXI4 read data channel signals for the slave port of the TLB block.

Table A-18 Read data channel signals-slave port of TLB block

AMBA equivalent	Slave port of TLB block		Direction
RID	**rid_s[I_S:0]**		Output
RDATA	**For 64-bit** **For 128-bit**	The data width is **rdata_s[63:0]** The data width is **rdata_s[127:0]**	Output
RRESP	**rresp_s[1:0]**		Output
RLAST	**rlast_s[0]**		Output
RVALID	**rvalid_s[0]**		Output
RUSER	**ruser_s[RUSER_WIDTH-1:0]**		Output
RREADY	**rready_s[0]**		Input

Table A-19 shows the AXI4 read data channel signals for the master port of the TLB block.

Table A-19 Read data channel signals-master port of TLB block

AMBA equivalent	Master port of TLB block	Direction
RID	**rid_m[I_M:0]**	Input
RDATA	**For 64-bit** The data width is **rdata_m[63:0]** **For 128-bit** The data width is **rdata_m[127:0]**	Input
RRESP	**rresp_m[1:0]**	Input
RLAST	**rlast_m[0]**	Input
RVALID	**rvalid_m[0]**	Input
RUSER	**ruser_m[RUSER_WIDTH-1:0]**	Input
RREADY	**rready_m[0]**	Output

Table A-20 shows the AXI4 read data channel signals for the master port of the PTW block.

Table A-20 Read data channel signals-master port of PTW block

AMBA equivalent	Master port of PTW block	Direction
RID	**rid_ptw[I_P:0]**	Input
RDATA	**For 64-bit** The data width is **rdata_ptw[63:0]** **For 128-bit** The data width is **rdata_ptw[127:0]**	Input
RRESP	**rresp_ptw[3:0]**	Input
RLAST	**rlast_ptw[0]**	Input
RVALID	**rvalid_ptw[0]**	Input
RUSER	-	Input
RREADY	**rready_ptw[0]**	Output

A.2.3 ACE-Lite signals

The following sections describe the ACE-Lite signals:

- *Write address channel signals* on page A-12.
- *Write data channel signals* on page A-12.
- *Write response channel signals* on page A-14.
- *Read address channel signals* on page A-14.
- *Read data channel signals* on page A-15.

See the *ARM ®CoreLink™ MMU-400 System Memory Management Unit AMBA® Designer (ADR-400) User Guide Supplement* for more information on the following ID widths:

- Master ID width, I_M.
- Slave ID width, I_S.
- PTW ID width, I_P.

Write address channel signals

Table A-21 shows the ACE-Lite write address channel signals.

Table A-21 Write address channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW block	Direction
AWID	**awid_s[I_S:0]**	Input	**awid_m[I_M:0]**	Output	**awid_ptw[I_P:0]**	Output
AWADDR	**awaddr_s[39:0]**	Input	**awaddr_m[39:0]**	Output	**awaddr_ptw[39:0]**	Output
AWLEN	**awlen_s[7:0]**	Input	**awlen_m[7:0]**	Output	**awlen_ptw[7:0]**	Output
AWSIZE	**awsize_s[2:0]**	Input	**awsize_m[2:0]**	Output	**awsize_ptw[2:0]**	Output
AWBURST	**awburst_s[1:0]**	Input	**awburst_m[1:0]**	Output	**awburst_ptw[1:0]**	Output
AWLOCK	**awlock_s[0]**	Input	**awlock_m[0]**	Output	**awlock_ptw[0]**	Output
AWCACHE	**awcache_s[3:0]**	Input	**awcache_m[3:0]**	Output	**awcache_ptw[3:0]**	Output
AWPROT	**awprot_s[2:0]**	Input	**awprot_m[2:0]**	Output	**awprot_ptw[2:0]**	Output
AWVALID	**awvalid_s[0]**	Input	**awvalid_m[0]**	Output	**awvalid_ptw[0]**	Output
AWREGION	**awregion_s[3:0]**	Input	**awregion_m[3:0]**	Output	**awregion_ptw[3:0]**	Output
AWQOS	**awqos_s[3:0]**	Input	**awqos_m[3:0]**	Output	**awqos_ptw[3:0]**	Output
AWSNOOP	**awsnoop_s[2:0]**	Input	**awsnoop_m[2:0]**	Output	**awsnoop_ptw[2:0]**	Output
AWBAR	**awbar_s[1:0]**	Input	**awbar_m[1:0]**	Output	**awbar_ptw[1:0]**	Output
AWDOMAIN	**awdomain_s[1:0]**	Input	**awdomain_m[1:0]**	Output	**awdomain_ptw[1:0]**	Output
AWUSER	**awuser_s[AWUSER_WIDTH+3:0]**	Input	**awuser_m[AWUSER_WIDTH+3:0]**	Output	-	-
AWREADY	**awready_s[0]**	Output	**awready_m[0]**	Input	**awready_ptw[0]**	Input

Note

For PTW, the write address channel signals are unused.

Write data channel signals

Table A-22 shows the ACE-Lite write data channel signals for the slave port of the TLB block.

Table A-22 Write data channel signals-slave port of TLB block

AMBA equivalent	Slave port of TLB block		Direction
WDATA	**For 64-bit**	The data width is **wdata_s[63:0]**	Input
	For 128-bit	The data width is **wdata_s[127:0]**	
WSTRB	**For 64-bit**	The data width is **wstrb_s[7:0]**	Input
	For 128-bit	The data width is **wstrb_s[15:0]**	
WLAST	**wlast_s[0]**		Input

Table A-22 Write data channel signals-slave port of TLB block (continued)

AMBA equivalent	Slave port of TLB block	Direction
WVALID	**wvalid_s[0]**	Input
WUSER	**wuser_s[WUSER_WIDTH-1:0]**	Input
WREADY	**wready_s[0]**	Output

Table A-23 shows the ACE-Lite write data channel signals for the master port of the TLB block.

Table A-23 Write data channel signals-master port of TLB block

AMBA equivalent	Master port of TLB block		Direction
WDATA	**For 64-bit** **For 128-bit**	The data width is **wdata_m[63:0]** The data width is **wdata_m[127:0]**	Output
WSTRB	**For 64-bit** **For 128-bit**	The data width is **wstrb_m[7:0]** The data width is **wstrb_m[15:0]**	Output
WLAST	**wlast_m[0]**		Output
WVALID	**wvalid_m[0]**		Output
WUSER	**wuser_m[WUSER_WIDTH-1:0]**		Output
WREADY	**wready_m[0]**		Input

Table A-24 shows the ACE-Lite write data channel signals for the master port of the PTW block.

Table A-24 Write data channel signals-master port of PTW

AMBA equivalent	Master port of PTW block		Direction
WDATA	**For 64-bit** **For 128-bit**	The data width is **wdata_ptw[63:0]** The data width is **wdata_ptw[127:0]**	Output
WSTRB	**For 64-bit** **For 128-bit**	The data width is **wstrb_ptw[7:0]** The data width is **wstrb_ptw[15:0]**	Output
WLAST	**wlast_ptw[0]**		Output
WVALID	**wvalid_ptw[0]**		Output
WUSER	-		-
WREADY	**wready_ptw[0]**		Input

Write response channel signals

Table A-25 shows the ACE-Lite write response channel signals.

Table A-25 Write response channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW block	Direction
BID	**bid_s[I_S:0]**	Input	**bid_m[I_M:0]**	Output	**bid_ptw[I_P:0]**	Output
BRESP	**bresp_s[1:0]**	Output	**bresp_m[1:0]**	Input	**bresp_ptw[1:0]**	Input
BVALID	**bvalid_s[0]**	Output	**bvalid_m[0]**	Input	**bvalid_ptw[0]**	Input
BUSER	**buser_s[BUSER_ WIDTH-1:0]**	Output	**buser_m[BUSER_ WIDTH-1:0]**	Input	-	-
BREADY	**bready_s[0]**	Input	**bready_m[0]**	Output	**bready_ptw[0]**	Output

Read address channel signals

Table A-26 shows the ACE-Lite read address channel signals.

Table A-26 Read address channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW block	Direction
ARID	**arid_s[I_S:0]**	Input	**arid_m[I_M:0]**	Output	**arid_ptw[I_P:0]**	Output
ARADDR	**araddr_s[39:0]**	Input	**araddr_m[39:0]**	Output	**araddr_ptw[39:0]**	Output
ARLEN	**arlen_s[7:0]**	Input	**arlen_m[7:0]**	Output	**arlen_ptw[7:0]**	Output
ARSIZE	**arsize_s[2:0]**	Input	**arsize_m[2:0]**	Output	**arsize_ptw[2:0]**	Output
ARBURST	**arburst_s[1:0]**	Input	**arburst_m[1:0]**	Output	**arburst_ptw[1:0]**	Output
ARLOCK	**arlock_s[0]**	Input	**arlock_m[0]**	Output	**arlock_ptw[0]**	Output
ARCACHE	**arcache_s[3:0]**	Input	**arcache_m[3:0]**	Output	**arcache_ptw[3:0]**	Output
ARPROT	**arprot_s[2:0]**	Input	**arprot_m[2:0]**	Output	**arprot_ptw[2:0]**	Output
ARVALID	**arvalid_s[0]**	Input	**arvalid_m[0]**	Output	**arvalid_ptw[0]**	Output
ARREGION	**arregion_s[3:0]**	Input	**arregion_m[3:0]**	Output	**arregion_ptw[3:0]**	Output
ARQOS	**arqos_s[3:0]**	Input	**arqos_m[3:0]**	Output	**arqos_ptw[3:0]**	Output
ARSNOOP	**arsnoop_s[3:0]**	Input	**arsnoop_m[3:0]**	Output	**arsnoop_ptw[3:0]**	Output
ARBAR	**arbar_s[1:0]**	Input	**arbar_m[1:0]**	Output	**arbar_ptw[1:0]**	Output
ARDOMAIN	**ardomain_s[1:0]**	Input	**ardomain_m[1:0]**	Output	**ardomain_ptw[1:0]**	Output
ARUSER	**aruser_s[ARUSER _WIDTH-1:0]**	Input	**aruser_m[ARUSER _WIDTH+3:0]**	Output	**aruser_ptw[3:0]**	Output
ARREADY	**arready_s[0]**	Output	**arready_m[0]**	Input	**arready_ptw[0]**	Input

Read data channel signals

Table A-27 shows the ACE-Lite read data channel signals for the slave port of the TLB block.

Table A-27 Read data channel signals-slave port of TLB block

AMBA equivalent	Slave port of TLB block		Direction
RID	**rid_s[I_S:0]**		Output
RDATA	**For 64-bit** **For 128-bit**	The data width is **rdata_s[63:0]** The data width is **rdata_s[127:0]**	Output
RRESP[a]	**rresp_s[1:0]**		Output
RLAST	**rlast_s[0]**		Output
RVALID	**rvalid_s[0]**		Output
RUSER	**ruser_s[RUSER_WIDTH-1:0]**		Output
RREADY	**rready_s[0]**		Input

a. In the ACE-Lite Specification, **RRESP** is 2 bits wide. However, when a shared interface is used in the MMU-400 for DVM operation, the ACE protocol definition has been used to include AC and CR signals. As a result, **RRESP** is increased in size by 2 bits that is [3:0]. Bits[3] and [2] are not on ACE-Lite interfaces, so you can tie **RRESP[3:2]** to `0x0`.

Table A-27 shows the ACE-Lite read data channel signals for the master port of the TLB block.

Table A-28 Read data channel signals-master port of TLB block

AMBA equivalent	Master port of TLB block		Direction
RID	**rid_m[I_M:0]**		Input
RDATA	**For 64-bit** **For 128-bit**	The data width is **rdata_m[63:0]** The data width is **rdata_m[127:0]**	Input
RRESP[a]	**rresp_m[1:0]**		Input
RLAST	**rlast_m[0]**		Input
RVALID	**rvalid_m[0]**		Input
RUSER	**ruser_m[RUSER_WIDTH-1:0]**		Input
RREADY	**rready_m[0]**		Output

a. In the ACE-Lite Specification, **RRESP** is 2 bits wide. However, when a shared interface is used in the MMU-400 for DVM operation, the ACE protocol definition has been used to include AC and CR signals. As a result, **RRESP** is increased in size by 2 bits that is [3:0]. Bits[3] and [2] are not on ACE-Lite interfaces, so you can tie **RRESP[3:2]** to `0x0`.

Table A-27 on page A-15 shows the ACE-Lite read data channel signals for the master port of the PTW block.

Table A-29 Read data channel signals-master port of PTW block

AMBA equivalent	Master port of PTW block	Direction
RID	**rid_ptw[I_P:0]**	Input
RDATA	**For 64-bit** The data width is **rdata_ptw[63:0]** **For 128-bit** The data width is **rdata_ptw[127:0]**	Input
RRESP[a]	**rresp_ptw[1:0]**	Input
RLAST	**rlast_ptw[0]**	Input
RVALID	**rvalid_ptw[0]**	Input
RUSER	-	-
RREADY	**rready_ptw[0]**	Output

a. In the ACE-Lite Specification, **RRESP** is 2 bits wide. However, when a shared interface is used in the MMU-400 for DVM operation, the ACE protocol definition has been used to include AC and CR signals. As a result, **RRESP** is increased in size by 2 bits that is [3:0]. Bits[3] and [2] are not on ACE-Lite interfaces, so you can tie **RRESP[3:2]** to `0x0`.

A.2.4 APB signals

This section describes the APB signals for the following:

- *APB4 signals*.
- *APB3 signals* on page A-17.

APB4 signals

Table A-30 shows the APB4 signals.

Table A-30 APB4 signals

AMBA equivalent	APB4 signals	Width	Direction
PADDR	**paddr**	32	Input
PWDATA	**pwdata**	32	Input
PSTROBE	**pstrobe**	4	Input
PPROT	**pprot**	3	Input
PWRITE	**pwrite**	1	Input
PENABLE	**penable**	1	Input
PSELx	**psel**	1	Input
PRDATA	**prdata**	32	Output
PSLVERR	**pslverr**	1	Output
PREADY	**pready**	1	Output
PCLKEN	**pclken**	1	Input

APB3 signals

Table A-31 shows the APB3 Secure and Non-secure signals.

Table A-31 APB3 signals

AMBA equivalent	Secure APB3 signals	Direction	Non-secure APB3 signals	Direction
PADDR	**paddr_s[31:0]**	Input	**paddr_ns[31:0]**	Input
PWDATA	**pwdata_s[31:0]**	Input	**pwdata_ns[31:0]**	Input
PWRITE	**pwrite_s**[0]	Input	**pwrite_ns**[0]	Input
PENABLE	**penable_s**[0]	Input	**penable_ns**[0]	Input
PSELx	**psel_s**[0]	Input	**psel_ns**[0]	Input
PRDATA	**prdata_s[31:0]**	Output	**prdata_ns[31:0]**	Output
PSLVERR	**pslverr_s**[0]	Output	**pslverr_ns**[0]	Output
PREADY	**pready_s**[0]	Output	**pready_ns**[0]	Output
PCLKEN	**pclken_s**[0]	Input	-	-

A.2.5 AXI low-power interface signals

Table A-32 shows the AXI low-power interface signals.

Table A-32 LPI signals

AXI LPI	TLB block	Direction	PTW block	Direction
CACTIVE	**cactive_tbu**	Output	**cactive_tcu**	Output
CSYSREQ	**csysreq_tbu**	Input	**csysreq_tcu**	Input
CSYSACK	**csysack_tbu**	Output	**csysack_tcu**	Output

A.2.6 Snoop channel signals

Table A-33 shows the ACE-Lite snoop channel signals.

Table A-33 Snoop channel signals

AMBA equivalent	Signal	Width	Direction	Description
Snoop address channel signals				
ACADDR	**acaddr_m**	40	Input	Snoop address
ACPROT	**acprot_m**	3	Input	Snoop protection information
ACVALID	**acvalid_m**	1	Input	Valid signal for the snoop address channel
ACSNOOP	**acsnoop_m**	4	Input	Snoop transaction type
ACREADY	**acready_m**	1	Output	Ready signal for the snoop address channel
Snoop response channel signals				

Table A-33 Snoop channel signals (continued)

AMBA equivalent	Signal	Width	Direction	Description
CRRESP	**crresp_m**	5	Output	Snoop response
CRVALID	**crvalid_m**	1	Output	Valid signal for the snoop response channel
CRREADY	**crready_m**	1	Input	Ready signal for the snoop response channel

A.3 Miscellaneous signals

This section describes the non-AMBA signals as follows:

- *Sideband signals*.
- *Interrupt signals*.
- *MBIST signals* on page A-20.
- *Authentication interface signal* on page A-20.
- *Tie-off signals* on page A-21.
- *Performance event signals* on page A-21.

A.3.1 Sideband signals

Table A-34 shows the sideband signals.

Table A-34 Sideband signals

Signal	Type	Width	Description
rsb_ns	Input	1	Determines the Non-secure state of an incoming read transaction. The value of this signal depends on the **arvalid** signal.
wsb_ns	Input	1	Determines the Non-secure state of an incoming write transaction. The value of this signal depends on the **awvalid** signal.
wsb_ssd	Input	0-8	Sideband signal to indicate the SSD index. If the **rsb_ns** or **wsb_ns** signal exists, then this signal does not exist. The value of this signal depends on the **awvalid** signal.
rsb_ssd	Input	0-8	Sideband signal to indicate the SSD index. If **the rsb_ns** or **wsb_ns** signal exists, then this signal does not exist. The value of this signal depends on the **arvalid** signal.
wsb_sid_s	Input	0-8	Sideband signal to indicate the write StreamID. The value of this signal depends on the **awvalid** signal.
rsb_sid_s	Input	0-8	Sideband signal to indicate the read StreamID. The value of this signal depends on the **arvalid** signal.

A.3.2 Interrupt signals

Table A-35 shows the interrupt signals generated by the MMU-400. See the *ARM® System Memory Management Unit Architecture Specification* for more information.

Table A-35 Interrupt signals

Signal	Type	Width	Description
cfg_flt_irpt_s	Output	1	Secure configuration access fault interrupt
cfg_flt_irpt_ns	Output	1	Non-secure configuration access fault interrupt
glbl_flt_irpt_s	Output	1	Global Secure fault interrupt
glbl_flt_irpt_ns	Output	1	Global Non-secure fault interrupt
prf_irpt	Output	1	Performance interrupt
cxt_irpt_ns	Output	1	Non-secure context interrupt
comb_irpt_ns	Output	1	Non-secure combined interrupt
comb_irpt_s	Output	1	Secure combined interrupt

A.3.3 MBIST signals

The MMU-400 supports a standard ARM MBIST interface to ensure that timing is met after inserting an MBIST multiplexer. ARM recommends no additional insertion of MBIST multiplexers. This interface exists only when the RAM option is selected for the TLB data. See the *ARM® CoreLink™ MMU-400 System Memory Management Unit Implementation Guide* for more information.

Table A-36 shows the MBIST signals.

Table A-36 MBIST signals

Signal	Type	Width	Description
mbistreq	Input	1	MBIST request from the MBIST controller to the TLB RAM.
mbistack	Output	1	Acknowlege from the MMU-400 that it is ready for an MBIST operation.
mbistaddr	Input	TLB_INDEX_WIDTH[a]-1:0	Right-justified address. This is the same as the physical address of the memory.
mbistreaden	Input	1	Read enable.
mbistwriteen	Input	1	Write enable.
mbistindata	Input	52	Write data.
mbistoutdata	Output	52	Read data, valid three clocks after read enable is set.

a. Where TLB_INDEX_WIDTH is equal to $\log_2$(TLB Depth).

A.3.4 Authentication interface signal

The authentication interface disables AXI accesses. Table A-37 shows the authentication interface signal. See the *ARM® CoreSight™ Architecture Specification*.

Table A-37 Authentication Interface signal

Signal	Type	Width	Description
spniden	Input	1	Secure privileged non-invasive debug enable. When the **spniden** signal is HIGH, it enables counting of Secure events. You can specify one of the following values: `0b0` Do not count Secure events in the performance counters. `0b1` Count Secure events in the performance counters.

A.3.5 Tie-off signals

Table A-38 shows the tie-off signals.

Table A-38 Tie-off signals

Signal	Type	Width	Description
cfg_cttw	Input	-	Indicates whether the MMU-400 supports coherent page table walks. This signal cannot change after reset.
testmode	Input	1	When **test_mode** is HIGH, the DFT testmode is enabled within the design. You can specify one of the following values: `0b0` Functional mode. `0b1` Test mode.

A.3.6 Performance event signals

Table A-39 shows the performance event signals.

Table A-39 Performance event signals

Signal	Type	Width	Description
event_clk	Output	1	Event counting every clock of the TLB block
event_clk64	Output	1	Event counting every 64th clock of the TLB block
event_wr_access	Output	1	Event counting every write access going through the TLB block
event_rd_access	Output	1	Event counting every read access going through the TLB block
event_wr_refill	Output	1	Event counting refill to the TLB as a result of a write access
event_rd_refill	Output	1	Event counting refill to the TLB as a result of a read access

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
Removed the configuration options and its range from Chapter 1 *Introduction*.	*Configurable options* on page 1-6	-
Added *Dynamic programming* on page 2-12 section.	Chapter 2 *Functional Description*	-
Added access privilege information in *About the programmers model* on page 3-2.	Chapter 3 *Programmers Model*	-
Added configuration and usage constraint attributes in all register description.	Chapter 3 *Programmers Model*	-
Updated the bit description for *Auxiliary Configuration Register bit assignments* on page 3-17.	Chapter 3 *Programmers Model*	-
Updated the reset values for Table 3-12 on page 3-22	Chapter 3 *Programmers Model*	-
Updated the data width for AXI 4 *Read data channel signals* on page A-10 and ACE-Lite™ *Read data channel signals* on page A-15.	Appendix A *Signal Descriptions*	-
Updated the width for **mbistaddr** in *MBIST signals* on page A-20.	Appendix A *Signal Descriptions*	-
Updated the note for **RRESP** in Table A-27 on page A-15.	Appendix A *Signal Descriptions*	-
Updated the *Authentication interface signal* on page A-20.	Appendix A *Signal Descriptions*	-
Updated Chapter 1 *Introduction*.	Chapter 1 *Introduction*	-

Table B-2 Differences between issue B and issue C

Change	Location	Affects
Added information on *About the MMU-400* on page 1-2 and *Features of the MMU-400* on page 1-4	Chapter 1 *Introduction*	All
Added information on *About the functions* on page 2-2 and *Interfaces* on page 2-3	Chapter 2 *Functional Description*	All
Added a note on *Fault handling* on page 2-11	Chapter 2 *Functional Description*	All
Removed the register description for the following registers: • Performance Monitor Counter Enable Set and Clear registers. • Performance Monitor Interrupt Enable Set and Clear registers. • Performance Monitor Overflow Status Set and Clear registers.	Chapter 3 *Programmers Model*	All