CoreLink™ MMU-400 System Memory Management Unit

Revision: r0p0

Technical Reference Manual

ARM®

ARM DDI 0472A (ID103111)

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	Initial release for r0p0

Proprietary Notice

Confidentiality Status

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

Product Status

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

Web Address

`http://www.arm.com`

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

Preface

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

About this book

This is the *Technical Reference Manual* 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 Description*

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 *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* on page vii
- *Timing diagrams* on page vii
- *Signals* on page vii.

Typographical

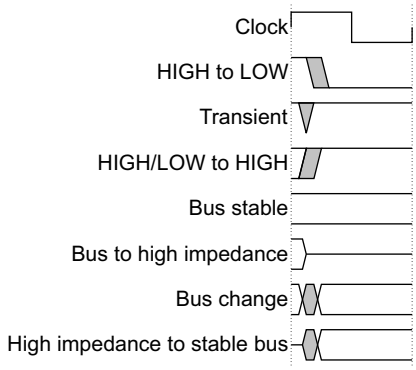
The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
< and >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`

Timing diagrams

The figure named *Key to timing diagram conventions* explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

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



Key to timing diagram conventions

Signals

The signal conventions are:

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

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

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:

- *CoreLink MMU-400 System Memory Management Unit Implementation Guide* (ARM DII 0265)
- *CoreLink MMU-400 System Memory Management Unit Integration Manual* (ARM DII 0266)
- *CoreLink MMU-400 System Memory Management Unit AMBA Designer (ADR-400) User Guide Supplement* (ARM DSU 0017)
- *ARM System Memory Management Unit Architecture Specification* (ARM IHI 0062)
- *AMBA® AXI Protocol Specification* (ARM IHI 0022)
- *CoreSight™ Architecture Specification* (ARM IHI 0029).

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 0472A
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

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

1.1 About the MMU-400

An MMU-400 controls address translation, access permissions, memory attribute determination, and checking at a memory system level. It provides a single stage of address translation on the accesses it receives on the AMBA channel, that is, AXI3, AXI4, or ACE-Lite. The following are the example masters for the MMU-400:

- *Graphics Processor Units* (GPUs)
- video engines
- *Direct Memory Access* (DMA) controllers
- *Color LCD* (CLCD) controllers
- network controllers.

An MMU-400 supports a single master interface. You can connect the master directly to the MMU-400, or multiple-masters can be connected through it, using N:1 interconnect logic in between.

The advantages of having an MMU-400, compared to direct physical addressing of the memory by devices, include:

- Large regions of memory are addressed by devices as a contiguous block without the requirement for the memory to be contiguous in physical memory.
- Memory protection from errant or misconfigured devices. A device cannot access memory that is not allocated to it.
- Under some virtualization schemes, guest *Operating Systems* (OSs) are not aware of the physical memory map or layout. Under such virtualization schemes, the Guest OS uses *Intermediate Physical Address* (IPA) address memory that is translated to *Physical Address* (PA) by the hypervisor. Without an MMU-400 to perform similar translation for the devices, most of the device driver code must migrate to the hypervisor. With an MMU-400, a guest OS can use the IPA to program a device.

The MMU-400 is set up to perform non-secure stage 2 address translations, and secure bypass translations.

1.2 The MMU-400 translation process

At the memory system level, an MMU-400 controls:

- security state determination
- address translation
- memory access permissions and determination of memory attributes
- memory attribute checks.

An access to an MMU-400 is referred to as a transaction.

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

1.3 Features of the MMU-400

The MMU-400 provides the following functionality:

- multiple transaction contexts that can apply to specific streams of transactions
- stage 2 translation
- fault handling, logging, and signaling
- debug and performance monitoring
- compatible with ARMv7 *Large Physical Address Extension* (LPAE) page table formats
- configurable depth write buffer
- Hit-Under-Miss (HUM) support
- the supported AXI data widths are 64 or 128 bits
- caching of page table entries in *Translation Look-aside Buffer* (TLB)
- supports the following protocols, when configured:
  - AXI3 protocol
  - AXI4 protocol
  - ACE-Lite with optional *Distributed Virtual Memory* (DVM) extensions.
- support for TrustZone extensions
- translation and protection checks.

1.4 Configurable options

This section describes the configuration options and the method of configuration. It contains the following sections:

- *MMU-400 configurability*
- *Configuration method* on page 1-8.

1.4.1 MMU-400 configurability

Table 1-1 shows the configuration options for the MMU-400. For more information, see the *CoreLink MMU-400 System Memory Management Unit Supplement to AMBA Designer (ADR-400) User Guide*.

Note

In this document, the term configured means statistically configured, where Table 1-1 defines a set of selected configuration options.

Table 1-1 Configuration options for the MMU-400

Configuration option	Range	Description
TLB Options		
AXI-ACE-Lite Protocol	ACE-Lite AXI3 AXI4	When a separate AXI master interface is selected for *Page Table Walk* (PTW), the separate master can have a different AXI type from the common AXI interface. If the same interface is used for PTW, then you must use the same AXI types. DVM signaling is available when the separate PTW or combined interface without separate PTW is of ACE-Lite type. AXI3 does not support: • write data interleaving • LOCK transactions.
AXI Data bus width	64 or 128	Width, in bits, of the AXI data bus. This is same for both AXI masters, if a separate PTW master is selected.
AXI ID signal width	0-23	The incoming AXI ID width is configurable from 0-23. The output AXI ID width changes based on the selected AXI type. For more information, see Table 1-2 on page 1-8.
TLB Depth	2-64	The TLB depth can 2-64.
Implement the TLB using a memory	RAM used for TLB RAM not used for TLB	You can implement the TLB data portion as RAM or flops. Implementing it as RAM optimizes area, but the setup and clock to Q delay for RAM is higher compared to using flops.
Depth of write buffer	0, 4, 8, or 16	The write buffer can accommodate multiple bursts up to the depth of the buffer. The write data path is not stalled for transactions that the write buffer can hold. For more information, see *HUM* on page 2-7.

Table 1-1 Configuration options for the MMU-400 (continued)

Configuration option	Range	Description
StreamID - Sideband signal width	0-8	The number of sideband signals used in the stream ID function. For more information, see *Generation of stream ID* on page 2-4.
StreamID - Includes WNR bit	Enable or Disable	You can enable this to include the R/W bit in the stream ID function. For more information, see *Generation of stream ID* on page 2-4.
StreamID - AXI ID bit selection	Selection of bits from AXI ID	Specifies which bits of the AXI_ID form part of the stream ID function. For more information, see *Generation of stream ID* on page 2-4.
Width of AXI slave interface AWUSER signals	0-128 bits	Width of AXI slave interface **AWUSER** signals.
Width of AXI slave interface WUSER signals	0-128 bits	Width of AXI slave interface **WUSER** signals.
Width of AXI slave interface BUSER signals	0-128 bits	Width of AXI slave interface **BUSER** signals.
Width of AXI slave interface ARUSER signals	0-128 bits	Width of AXI slave interface **ARUSER** signals.
Width of AXI slave interface RUSER signals	0-128 bits	Width of AXI slave interface **RUSER** signals.
PTW Options		
Number of Contexts	1-8	Specifies the number of parallel contexts supported for a page table walk. You can specify this many page table base addresses and corresponding attributes. The actual number of page table walks that can occur in parallel is fixed at 4.
Num of SMRs	2, 4, 8, 16, 24, or 32	The number of stream mapping registers. ARM recommends that the number of SMRs selected are not greater than $2x2^{(STREAM\_ID\_WIDTH)}$, because those registers are not used.
PTW has separate AXI/ACE port	Enable or Disable	You can select a separate AXI interface for PTW. This ensures the other AXI is reserved for device transactions only.
AXI/ACE-Lite Protocol	ACE-Lite AXI3 AXI4	Specifies the protocol of the separate AXI port on the PTW block.
APB3/4 Protocol	APB3 or APB4	When you select APB3, two independent APBs are generated, one for secure and one for non-secure transactions. Secure APB has priority over non-secure APB. When you select APB4, one APB interface is generated. The **PPROT[1]** signal differentiates between secure and non-secure transactions.

Table 1-1 Configuration options for the MMU-400 (continued)

Configuration option	Range	Description
Security		
Use SSD Determination Table	SSD table present SSD table not present	When you select the option as *Security State Determination* (SSD) table not present, then the non-secure state is directly assigned to the incoming sideband signals, along with the transaction. **Writes** Non-secure state = wsb_ns, where wsb_ns is the write sideband signal for security **Reads** Non-secure state = rsb_ns, where rsb_ns is the read sideband signal for security When you select the option as SSD table present, then you must program the security state in the table.
SSD Sideband signal width	0-8	If the non-secure state is not directly assigned to incoming sideband signals, then the SSD index is selected as a function of the incoming AXI ID and sideband signals. For more information, see *Generation of the SSD index* on page 2-5.
SSD - AXI ID bit selection	Selection of bit from AXI ID	Specifies which bits of the AXI_ID are included in the SSD index function. For more information, see *Generation of the SSD index* on page 2-5.
Specify Use of SSDIndex0-31 Specify SSDIndex0-31	Disable Secure Prog-Secure Prog-NonSecure	Specifies which entries in the SSD table are secure. For more information, see *Generation of the SSD index* on page 2-5.
Registering		
AW slave interface registering options W slave interface registering options B slave interface registering options AR slave interface registering options R slave interface registering options	*Forward* (Fwd) registered *Reverse* (Rev) registered Full registered Bypass registered	Each AXI channel has a configurable register slice in the MMU-400 slave interface. This register slice is configured to be bypass, fully registered, forward registered, or reverse registered. An I/O delay of 70 percent of the clock is assumed for interfaces that are driven to or driven by a register. An I/O delay of 40 percent of the clock is assumed for interfaces that are bypassed.
Register the bus between PTW and TLB blocks Register the bus between TLB and PTW blocks	*Forward* (Fwd) registered *Reverse* (Rev) registered Full registered Bypass registered	Register slice for PTW request and acknowledge bus. A register slice on the bus which carries information about the PTW request and acknowledge. This can help if the routing delay between the TLB and the PTW logic is high.

Table 1-2 shows the AXI types and suitable input and output AXI ID widths and values for TLB block.

Table 1-2 The MMU-400 top configurability for TLB block

Transaction	AXI type	Width		Value	
		Input AXI ID, I_S	Output AXI ID, I_M	Input AXI ID	Output AXI ID
Passing through incoming transactions on the TLB block with address translation	ACE-Lite	0	4	-	4'b0000
		1	4	[x]	{3'b000,[x]}
		2	4	[x]	{2'b00,[x]}
		3-23	IDW+1	[x]	{1'b0,[x]}
Sync transaction generated by the TLB block	ACE-Lite	0-2	4	-	4'b1111
		3-23	IDW+1	-	{IDW+1{1'b1}}
Passing through incoming transaction on the TLB block, with Address translation	AXI3 or AXI4	0	3	-	3'b000
		1	3	[x]	{2'b00,[x]}
		2-23	IDW+1	[x]	{1'b0,[x]}

Table 1-3 shows the AXI types and suitable input and output AXI ID widths and values for PTW block.

Table 1-3 The MMU-400 top configurability for PTW block

Transaction	AXI type	Width	Value	
		Output AXI ID, I_P	Input AXI ID	Output AXI ID
PTW transaction generated by the PTW block	ACE-Lite	4	-	4'b1000 – 4'b1011
		IDW+1	-	{2'b10,{IDW-3{1'b0}},2'b00} – {2'b10,{IDW-3{1'b0}},2'b11}
Sync complete transaction generated by the PTW block	AXI3 or AXI4	4	-	4'b1100
		IDW+1	-	{3'b110, {IDW-2{1'b0}}}
PTW transaction generated by the PTW block	AXI3 or AXI4	3	-	3'b100 – 3'b111
		IDW+1	-	{1'b1,{IDW-2{1'b0}},2'b00} – {1'b1,{IDW-2{1'b0}},2'b11}

1.4.2 Configuration method

The format of the MMU-400 output is an XSL template. The AMBA Designer GUI is used for the MMU-400 configuration. The configuration options are used to render the XSL templates into Verilog. See *Configurable options* on page 1-5.

See the *CoreLink MMU-400 System Memory Management Unit AMBA Designer (ADR-400) User Guide Supplement*.

1.5 Product revisions

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

r0p0 First release.

Chapter 2
Functional Description

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

2.1 About programming interfaces

The MMU-400 requires a programming interface to permit software to configure the controller and to initialize the memory devices. The following sections discuss the MMU-400 programming interfaces and their differences:

- *APB interface*
- *Address map*
- *AXI3 support*
- *AXI4 support* on page 2-3.

2.1.1 APB interface

The MMU-400 provides either one of the following:

- The APB4 programming interface. For more information about integrating into AMBA4 systems, see the *AMBA 4.0 Specification*.
- A pair of APB3 programming interfaces. One of the APB interfaces is indicated as secure, and other interface is indicated as non-secure. For more information, see the *AMBA 3 APB Protocol Specification*.

2.1.2 Address map

The address map of the programming interface is consistent with the *ARM System Memory Management Unit Architecture Specification*.

The following IMPLEMENTATION DEFINED registers are implemented in the MMU-400, in addition to the registers specified in the *ARM System Memory Management Unit Architecture Specification*:

- Integration Mode Register, ITCTRL
- Integration Input Register, ITIP
- Integration Output Register, ITOP.

For more information about these registers and bit descriptions, see *Integration registers* on page 3-33.

2.1.3 AXI3 support

The MMU-400 TLB block supports the AXI3 protocol, when configured to do so. In this mode, only AXI3 signaling is present on the main data path through the MMU-400.

DVM is only supported, if a separate ACE-Lite master port is configured for the PTW block.

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

- Write data interleaving.

  Note

  Data corruption can occur if write data and write address ordering are not the same.

- Locked transactions, you cannot set **AxLOCK[1]** 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 in the **AxLOCK** signal are supported.

  Note

  If the MMU-400 receives a lock transaction, the output transaction is presented as a normal access.

2.1.4 AXI4 support

The MMU-400 TLB block supports the AXI4 protocol, when configured to do so. In this mode, only AXI4 signaling is present on the main data path through the MMU-400.

DVM is only supported if a separate ACE-Lite master port is configured for the PTW block.

2.1.5 ACE-Lite support

The MMU-400 TLB block supports the ACE-Lite protocol, when configured to do so. In this mode, only ACE-Lite signaling is present on the main data path through the MMU-400.

In this mode, the DVM is supported with a combined master port, or when a separate ACE-Lite master port is configured for the PTW block.

2.2 Generation of stream ID

The stream ID is selected during the configuration. It is selected as a function of the incoming AXI ID, a read or a write transaction, and up to 8 sideband signals. The maximum possible number of bits that you can select is 1-15.

AWID, ARID Bit[22:0]. If the ID width is less than 23 bits, then you can set only the valid bits to 1.

Read or Write Bit[23]. If this bit is set, then generate the stream ID with this bit set to 1 for writes, and this bit set to 0 for reads.

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

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

Example 2-1 Generation of stream ID - example 1

If bits 7, 4, 3, and 2 of AWID/ARID, R/W bit, and 4 sideband signals are selected to generate the stream ID, then the final generated stream ID has the following format:

Writes `Stream_id = {wsb_sid[3:0],1'b1,awid[7],awid[4],awid[3],awid[2]}`

Reads `Stream_id = {rsb_sid[3:0],1'b0,arid[7],arid[4],arid[3],arid[2]}`

Example 2-2 Generation of stream ID - example 2

If bits 2, 1, and 0 of AWID/ARID are selected to generate the stream ID, that is, no R/W bit and no sideband signals, then the final generated stream ID has the following format:

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 side band signals for the stream ID respectively
- **awid** and **arid** are the write and read ID signals respectively.

2.3 Generation of the SSD index

The generation of the SSD index is similar to that of the stream ID.

The SSD index is selected during the configuration. It is selected as a function of the incoming AXI ID, and up to eight sideband signals. The maximum possible number of bits that you can select is 0-15.

AWID/ARID Bit[22:0]. If ID width is less than 23 bits, then you can set the valid bits to 1.

Sideband signals You can have a maximum eight sideband signals.

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

Example 2-3 Generation of stream ID - example 1

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

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]}`

Example 2-4 Generation of stream ID - example 2

If bits 13,10, and 5 of **AWID**/**ARID** are selected to generate the SSD index, that is, no sideband signals, then the final generated SSD index has the following format:

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.

2.4 Determining the security state of masters

When the SSD index is determined, the SSD table comprises bits from 0-2$^{(\text{SSD Index WIDTH})-1}$. You must determine the status of all 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 supply the indices of those masters whose security state are always secure.

List of programmable indices

You can programme the security state of these indices.

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

Indices that are not included in non-programmable secure and programmable are assumed to be non-programmable non-secure indices.

Note

You must specify each entry in one list and each list can be empty. At least one programmable or fixed non-secure entry must be available in every configuration.

Example 2-5 Programmable and non-programmable indices

If the SSD index width is 6 bits, then there are 64 indices, whose security states must be determined. The following are few examples of the indices:

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 not listed from 0-63 are fixed non-secure indices.

2.5 HUM

HUM functionality, enabled by the write buffer, enables the hit transactions coming after miss transactions to be translated by the MMU-400. Hit transactions get translated only if the write data from the miss transaction can be accommodated in the write buffer.

HUM for writes is automatically disabled when the write buffer depth is 0, and is automatically enabled when the write buffer depth is not 0. HUM for reads is always ON, and is not affected by the write buffer depth.

The number of outstanding missed transactions is determined by the depth of the write buffer. For example, if the depth of the buffer is 4 then it can hold two transactions of length 2. Each buffer entry holds only one beat of the transaction, even if it is of a narrow width.

Example 2-6 Hit under miss

If you have selected a write buffer depth of 8, and there is a missed transaction of length 4 and another missed transaction of length 3, then both these transactions are stored in the write buffer, while the page table walks for these transactions is being performed.

If another transaction comes in that is a hit, then that transaction would be permitted to pass through.

Note

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

2.6 Fault handling

An MMU-400 only supports the terminate fault handling mode as *ARM System Memory Management Unit Architecture Specification* describes. The fault stalling is not supported.

On fault, the faulted transactions results in 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 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.

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

Chapter 3
Programmers Model

This chapter describes the MMU-400 registers and provides information about programming the MMU-400. It contain 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

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.

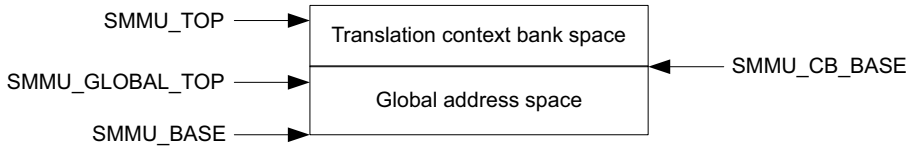


Figure 3-1 The MMU-400 address map, global space and translation context bank

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 map

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* describes:

- *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	0x0	*Secure Configuration Register 0* on page 3-10	Banked with security
SMMU_SCR1	RW	S	0x4	Secure Configuration Register 1	Secure only
SMMU_ACR	RW	NS	0x10	*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	0x20	*Identification registers* on page 3-17	-
SMMU_IDR1		S NS	0x24		-
SMMU_IDR2		S NS	0x28		-
SMMU_IDR7		S NS	0x3C		-
SMMU_SGFAR[31:0] SMMU_GFAR[31:0]	RW	S	0x40	Global Fault Address Register	Banked with security
SMMU_SGFAR[63:32] SMMU_GFAR[63:32]			0x44		
SMMU_GFSR	RW	NS	0x48	Global Fault Status Register	-
SMMU_SGFSR		S			Banked with security
SMMU_GFSRRESTORE	WO	NS	0x4C	Global Fault Status Register Restore	-
SMMU_SGFSRRESTORE		S			Banked with security
SMMU_GFSYNR0	RW	NS	0x50	Global Fault Syndrome Register 0	-
SMMU_SGFSYNR0		S			Banked with security
SMMU_GFSYNR1	RW	NS	0x54	Global Fault Syndrome Register 1	-
SMMU_SGFSYNR1		S			Banked with security
SMMU_STLBIALL	WO	S NS	0x60	Invalidate entire TLB Register	Secure only
SMMU_TLBIVMID	WO	NS	0x64	Invalidate TLB by VMID Register	-
SMMU_TLBIALLNSNH	WO	NS	0x68	Invalidate Entire Non-secure Non-Hyp TLB Register	-
SMMU_STLBGSYNC	WO	S NS	0x70	Global Synchronize TLB Invalidate Register	-
SMMU_TLBGSYNC					-
SMMU_STLBGSTATUS	RO	S NS	0x74	Global TLB Status Register	Banked with security
SMMU_TLBGSTATUS					-
SMMU_DBGRPTR	RW	S	0x80	*Debug registers* on page 3-23	-
SMMU_DBGRDATA	RO	S	0x84		-
SMMU_NSCR0	RW	S	0x400	*Secure Alias to Non-secure Configuration Register 0* on page 3-27	Secure only
SMMU_NSACR	RW	S	0x410	*Secure Alias to Non-secure Auxiliary Configuration Register* on page 3-27	Secure only

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

Name	Type	S or NS[a]	Offset	Description	Notes
SMMU_NSGFAR[31:0][b]	RW	S	0x440	Secure alias for Non-secure Global Fault Address Register	Secure only, 64-bit
SMMU_NSGFAR[63:32][b]	RW	S	0x444	Secure alias for Non-secure Global Fault Address Register	
SMMU_NSGFSR[b]	WO	S	0x448	Secure alias for Non-secure Global Fault Status Register	Secure only
SMMU_NSGFSRRESTORE[b]	WO	S	0x44C	Secure alias for Non-secure Global Fault Status Register Restore	Secure only
SMMU_NSGFSYNR0[b]	RW	S	0x450	Secure alias for Non-secure Global Fault Syndrome Register 0	Secure only
SMMU_NSGFSYNR1[b]	RW	S	0x454	Secure alias for Non-secure Global Fault Syndrome Register 1	Secure only
SMMU_NSTLBGSYNC[b]	WO	S	0x470	Secure alias for Non-secure Global Synchronize TLB Invalidate Register	Secure only
SMMU_NSTLBGSTATUS[b]	RO	S	0x474	Secure alias for Non-secure Global TLB Status Register	Secure only
SMMU_SMR*n*	RW	S NS	0x800-0x87C	*Stream Match registers* on page 3-27	-
SMMU_S2CR*n*	RW	S NS	0xC00 - 0xC7C	*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
ITCTRL	RW	NS/S	0x2000	*Integration Mode Control Register* on page 3-33
ITOP	RW	NS/S	0x2004	*Integration Test Output Register* on page 3-34
ITIP	RO	NS/S	0x2008	*Integration Test Input Register* on page 3-33

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* on page 3-39
PMCNTENCLR*n*		NS	0x3C20	
PMINTENSET*n*	RW	NS	0x3C40	*Performance Monitor Interrupt Enable Set and Clear registers* on page 3-40
PMINTENCLR*n*			0x3C60	
PMOVSCLR*n*	RW	NS	0x3C80	*Performance Monitor Overflow Status Set and Clear registers* on page 3-41
PMOVSSET*n*			0x3CC0	
PMCFGR	RO	NS	0x3E00	*Performance Monitor Configuration Register* on page 3-42
PMCR	RW	NS	0x3E04	*Performance Monitor Control Register* on page 3-43
PMCEID0-1	RO	NS	0x3E20-0x3E24	Performance Monitor Common Event ID Register
PMAUTHSTATUS	RO	NS	0x3FB8	*Performance Monitor Authentication Status Register* on page 3-45
PMDEVTYPE	RO	NS	0x3FCC	*Performance Monitor Device Type Register* on page 3-46

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	0x4000-0x4FFC	*The MMU-400 Security State Determination Address Space* on page 3-47

a. If SSD table is not implemented, the bits are UNK/SBOP.
If 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	0xFFD0	*Peripheral Identification registers* on page 3-48
Periph ID 5	RO	NS/S	0xFFD4	
Periph ID 6	RO	NS/S	0xFFD8	
Periph ID 7	RO	NS/S	0xFFDC	
Periph ID 0	RO	NS/S	0xFFE0	
Periph ID 1	RO	NS/S	0xFFE4	
Periph ID 2	RO	NS/S	0xFFE8	
Periph ID 3	RO	NS/S	0xFFEC	
Component ID0	RO	NS/S	0xFFF0	*Component Identification registers* on page 3-48
Component ID1	RO	NS/S	0xFFF4	
Component ID2	RO	NS/S	0xFFF8	
Component ID3	RO	NS/S	0xFFFC	

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	0x0	*System Control Register* on page 3-52
SMMU_CB*n*_TTBR0[64:0]	RW	64	0x20-0x24	Translation Table Base Register
SMMU_CB*n*_TTBCR	RW	32	0x30	*Translation Table Base Control Register* on page 3-55
SMMU_CB*n*_FSR[a]	-	-	0x58	Fault Status Register. See Table 3-1 on page 3-4
SMMU_CB*n*_FSRRESTORE[a]	-	-	0x5C	Fault Status Restore Register. See Table 3-1 on page 3-4

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

Name	Type	Size	Offset	Description
SMMU_CB*n*_FAR[31:0][a]	-	-	0x60	Fault Address Register. See Table 3-1 on page 3-4
SMMU_CB*n*_FAR[63:32][a]	-	-	0x64	
SMMU_CB*n*_FSYNR0[a]	-	-	0x68	Fault Syndrome Registers. See Table 3-1 on page 3-4
SMMU_CB*n*_PMXEVCNTR*m*	-	-	0xE00-0xE08	Performance Monitor Event Counter registers. See *Performance Monitoring registers* on page 3-36
SMMU_CB*n*_PMXEVTYPER*m*	-	-	0xE80-0xE88	Performance Monitor Event Type Registers. See *Performance Monitoring registers* on page 3-36
SMMU_CB*n*_PMCFGR	-	-	0xF00	Performance Monitor Configuration Register. See *Performance Monitoring registers* on page 3-36
SMMU_CB*n*_PMCR	-	-	0xF04	Performance Monitor Control Register. See *Performance Monitoring registers* on page 3-36
SMMU_CB*n*_PMCEID0-1	-	-	0xF20	Performance Monitor Common Event Identification registers. See *Performance Monitoring registers* on page 3-36
SMMU_CB*n*_PMCNTENSET	-	-	0xF40	Performance Monitor Count Enable Set Register. See *Performance Monitoring registers* on page 3-36
SMMU_CB*n*_PMCNTENCLR	-	-	0xF44	Performance Monitor Count Enable Clear Register. See *Performance Monitoring registers* on page 3-36
SMMU_CB*n*_PMINTENSET	-	-	0xF48	Performance Monitor Interrupt Enable Set Register. See *Performance Monitoring registers* on page 3-36
SMMU_CB*n*_PMINTENCLR	-	-	0xF4C	Performance Monitor Interrupt Enable Clear Register. See *Performance Monitoring registers* on page 3-36
SMMU_CB*n*_PMOVSRCLR	-	-	0xF54	Performance Monitor Overflow Status Clear Register. See *Performance Monitoring registers* on page 3-36
SMMU_CB*n*_PMOVSRSET	-	-	0xF50	Performance Monitor Overflow Status Set Register. See *Performance Monitoring registers* on page 3-36
SMMU_CB*n*_PMAUTHSTATUS	-	-	0xFB8	Performance Monitor Authentication Status Register. See *Performance Monitoring registers* on page 3-36

a. Follow the same format as *The MMU-400 Global Register Space 0* on page 3-10 describes.

3.4 The MMU-400 Global Register Space 0

The MMU-400 Global Register Space 0 gives 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 Secure Configuration Register 0 characteristics are:

Purpose The Secure Configuration Register, SMMU_SCR0, contains top-level control of the MMU-400, that is, only accessible by secure access.

Note

The Non-secure register, SMMU-CR0, does not provide full top-level control of the MMU-400 for secure transactions. In implementations supporting the security extensions, some SMMU_CR0 fields only apply to Non-secure transactions.

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

Figure 3-2 on page 3-11 shows the Secure Configuration Register 0 bit assignments.

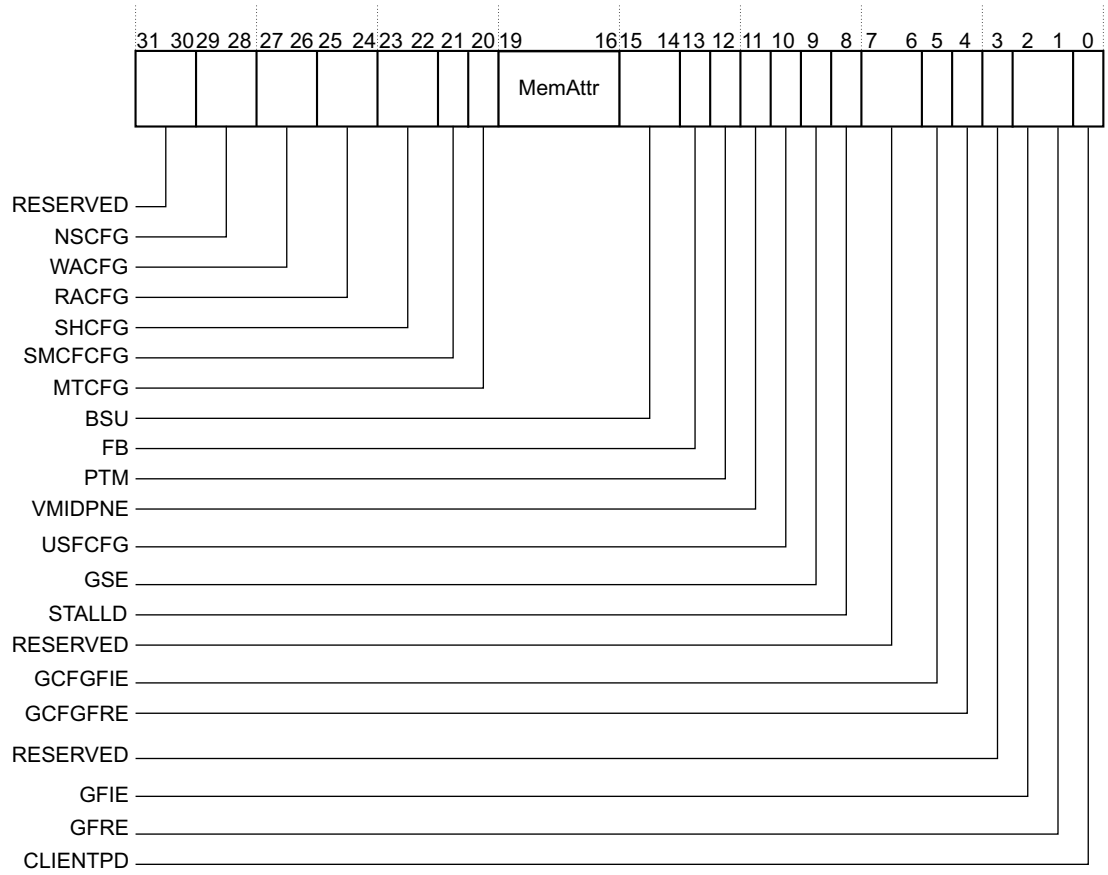


Figure 3-2 Secure Configuration Register 0 bit assignments

Table 3-8 shows the Secure Configuration Register 0 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. **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. The encodings of this field are: 00 Use APROTNS as presented with the transaction. 01 RESERVED. 10 Secure. 11 Non-secure.
[27:26]	WACFG	-	Write Allocate Configuration. Controls the allocation hint for write accesses. The encodings of this field are: 00 Use allocation attributes as presented with the transaction. 01 RESERVED. 10 Allocate. 11 No allocate. **Note** This field applies to transactions that bypass the MMU-400 stream mapping table. 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: 00 Use allocation attributes as presented with the transaction. 01 RESERVED. 10 Allocate. 11 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: 00 Use shareable attribute as presented with the transaction. 01 Outer-shareable. 10 Inner-shareable. 11 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 pass 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 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: `00` No effect. `01` Inner shareable. `10` Outer shareable. `11` 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.BTM. 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 permitted to affect cached translation in the MMU-400, and is permitted to apply to all unlocked entries. In implementations supporting the 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 not coordinated with the wider system. If VMIDPNE has the value 1, broadcast TLB Invalidate operations that specify a VMID value are not required to apply to cached translations in the MMU-400. This field is a hint. A broadcast TLB Invalidate operation is still permitted to affect cached translation in the MMU-400 and is permitted to 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 supporting the 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 is permitted to 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 global configuration fault. 1 Raise an interrupt on 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 global configuration fault. 1 Return an abort on 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 global fault. 1 Raise an interrupt on global fault.
[1]	GFRE	0	Global Fault Report Enable. The possible values of this bit are: 0 Do not return an abort on global fault. 1 Return an abort on 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 Auxiliary Configuration Register characteristics are:

Purpose For the MMU-400, the Auxiliary Configuration Register, SMMU_ACR and SMMU_SACR, are defined as Table 3-9 on page 3-17 shows.

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

Figure 3-3 shows the Auxiliary Configuration Register bit assignments.

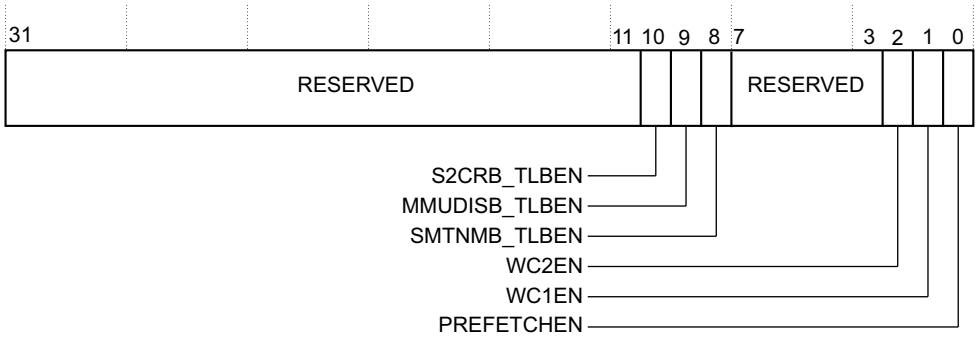


Figure 3-3 Auxiliary Configuration Register bit assignments

Note

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

Table 3-9 shows the Auxiliary Configuration Register 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 TLB with the S2CR bypassed transaction details. 1 Update TLB with the S2CR bypassed transaction details.
[9]	MMUDISB_TLBEN	MMU Disabled Bypass TLB Enabled. The possible values of this bit are: 0 Do not Update TLB with the MMU-400 disabled transaction details. 1 Update 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 TLB with the Stream Match Table No Match bypassed transaction details. 1 Update TLB with the Stream Match Table No Match bypassed transaction details.
[7:3]	RESERVED	RESERVED.
[2]	WC2EN	Walk Cache 2 Enabled. The possible values of this bit are: 0 Disables the walk cache 2. 1 Enables the walk cache 2.
[1[	WC1EN	Walk Cache 1 Enabled. The possible values of this bit are: 0 Disables the walk cache 1. 1 Enables the walk cache 1.
[0]	PREFETCHEN	Pre-fetch buffer Enabled. 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 Identification Register characteristics 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 3-23.

The MMU-400 provides facilities to permit the secure software to reserve some MMU-400 resources for its own use. See *Identification Register 0* on page 3-18.

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.

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

Identification Register 0

Figure 3-4 shows the Identification Register 0 bit assignments.

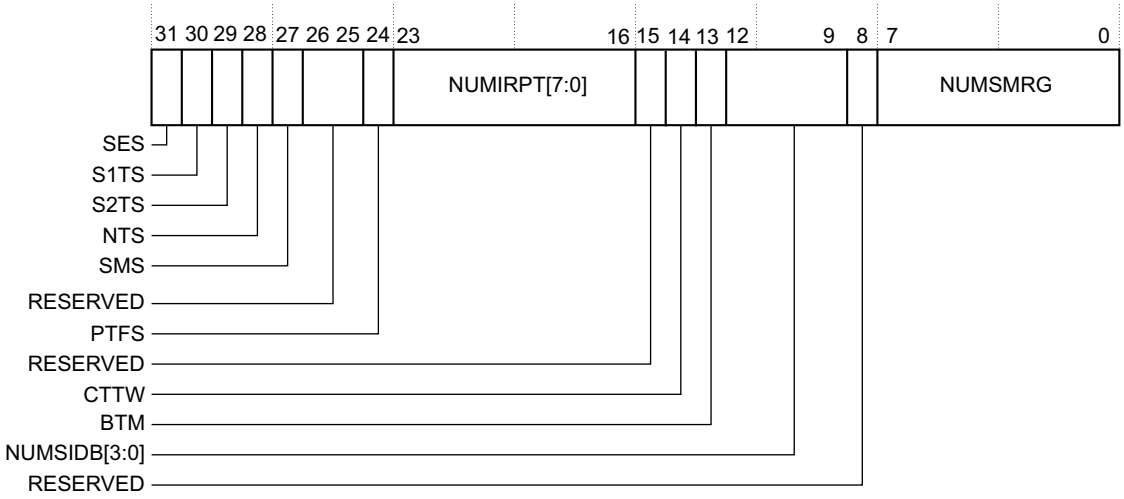


Figure 3-4 Identification Register 0 bit assignments

Table 3-10 shows the Identification Register 0 bit assignments.

Table 3-10 Identification Register 0 bit assignments

Bits	Name	Reset value	Description
[31]	SES	1	Security Extension Support.
[30]	S1TS	0	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.
[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.

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

Bits	Name	Reset value	Description
[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 supporting the 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 Stream ID width	Number of Stream ID bits: NUMSIDB[3:0] indicates the number of Stream ID bits that are implemented. The valid range is 0-15. **Note** You can anticipate a 0 bit Stream ID, where the sources 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 Match table. See Table 1-1 on page 1-5 for valid values for the number of stream mapping registers. In implementations supporting the Stream Match function, this field has a value of greater than or equal to 1. **Interaction with security extensions** In implementations supporting the security extensions, the SMMU_IDR0 version of this field reflects the configured value in SMMU_SCR1.NSNUMSMRGO. **Note** In implementations supporting the Stream Match function, the secure software must provide the non-secure software with the use of at least one Stream Match register group.

Identification Register 1

Figure 3-5 shows the Identification Register 1 bit assignments.

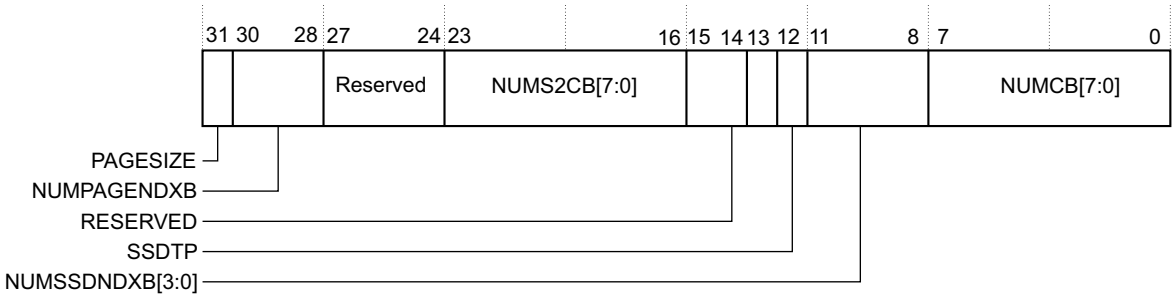


Figure 3-5 Identification Register 1 bit assignments

Table 3-11 shows the Identification Register 1 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 format. This field is validated by SMMU_IDR0.S2TS. See Table 1-1 on page 1-5 for valid values for number of valid contexts.
[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* for more information.
[14:13]	RESERVED	-	RESERVED.

Table 3-11 Identification Register 1 bit assignments (continued)

Bits	Name	Reset value	Description
[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 the 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 the 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 3-52. The value reported in NUMCB includes Translation Context Banks that only support the stage 2 format. **Interaction with security extensions** In implementations supporting the security extensions, the non-secure reads of SMMU_IDR1 reflects the configured value in SMMU_SCR1.NSNUMCBO.

Identification Register 2

Figure 3-6 shows the Identification Register 2 bit assignments.

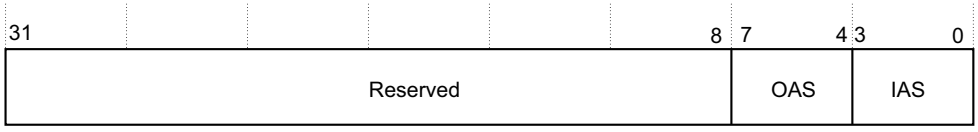


Figure 3-6 Identification Register 2 bit assignments

Table 3-12 shows the Identification Register 2 bit assignments.

Table 3-12 Identification Register 2 bit assignments

Bits	Name	Reset value	Description
[31:8]	RESERVED	-	RESERVED.
[7:4]	OAS	0	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	0	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 Identification Register 7 bit assignments.

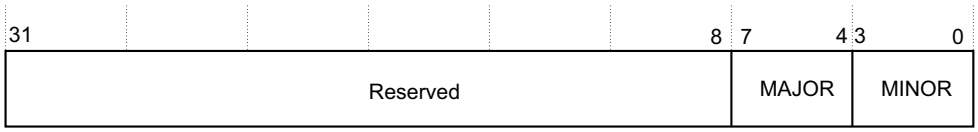


Figure 3-7 Identification Register 7 bit assignments

Table 3-13 shows the Identification Register 7 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

The MMU-400 supports TLB Visibility by providing a read pointer register, and a read data register to read the value. The Read Pointer register is initialized at reset to 0, and is auto-incremented by 1 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 2 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 happen when there are no outstanding transactions. If TLB reads happen in conjunction with transactions, then the TLB read can return data before or after it has been updated.

SMMU_DBGRPTR, Debug Read Pointer Register

For the MMU-400, bits [23:16] are always 0 unless written specifically by an APB access.

Figure 3-8 shows the MMU-400 Debug Read Pointer Register bit assignments.

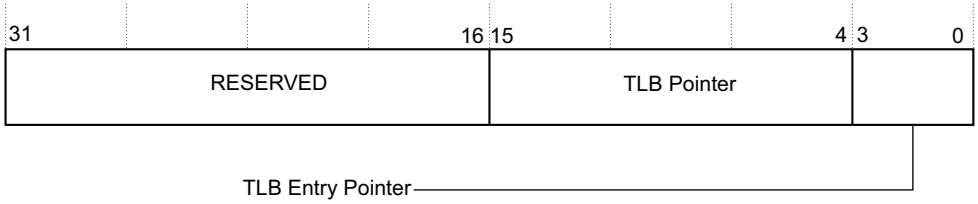


Figure 3-8 Debug Read Pointer Register bit assignments

Table 3-14 shows the MMU-400 Debug Read Pointer Register 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

Table 3-15 shows the data format of Debug Read Data Register.

Table 3-15 Debug Read Data Register data format

Bits	Width	Description
Word 1		
[31:4]	28	Virtual Address used for address lookup.
[3:2]	2	The encodings of this field are: `2b00` Word which is not first or last of the TLB entry. `2b01` First word for that TLB entry. `2b10` Last word for that TLB entry. `2b11` First word for that TLB.

Table 3-15 Debug Read Data Register data format (continued)

Bits	Width	Description
[1]	1	The possible values of this bit are: `0` Pointer is valid. `1` Pointer is invalid.
[0]	1	The possible values of this bit are: `0` Entry is valid. `1` Entry is invalid.
Word 2		
[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 Attributes. See the *ARM System Memory Management Unit Architecture Specification*.
[7]	1	RESERVED.
[6:4]	3	Page Size. The encodings of this field are: `3b000` 4KB. `3b001` RESERVED. `3b010` RESERVED. `3b011` 2MB. `3b100` RESERVED. `3b101` 1GB. `3b110` RESERVED. `3b111` RESERVED.
[3:2]	2	The encodings of this field are: `2b00` Word which is not first or last of the TLB entry. `2b01` First word for that TLB entry. `2b10` Last word for that TLB entry. `2b11` First word for that TLB.
[1]	1	The possible values of this bit are: `0` Pointer is valid. `1` Pointer is invalid.

Table 3-15 Debug Read Data Register data format (continued)

Bits	Width	Description
[0]	1	The possible values of this bit are: `0` Entry is valid. `1` Entry is invalid.
Word 3		
[31:17]	15	Stream ID.
[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	The encodings of this field are: `2b00` Word which is not first or last of the TLB entry. `2b01` First word for that TLB entry. `2b10` Last word for that TLB entry. `2b11` First word for that TLB.
[1]	1	The possible values of this bit are: `0` Pointer is valid. `1` Pointer is invalid.
[0]	1	The possible values of this bit are: `0` Entry is valid. `1` Entry is invalid.
Word 4		
[31:17]	15	Stream ID Mask.
[16:13]	4	RESERVED.
[12:11]	2	Entry Type. The encodings of this field are: `2b00` Translation is enabled. `2b01` Translation is disabled. `2b10` S2CR bypass as programmed in the S2CR Register. `2b11` 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*.
[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	The encodings of this field are: `2b00` Word which is not first or last of the TLB entry. `2b01` First word for that TLB entry. `2b10` Last word for that TLB entry. `2b11` First word for that TLB.

Table 3-15 Debug Read Data Register data format (continued)

Bits	Width	Description
[1]	1	The possible values of this bit are: `0` Pointer is valid. `1` Pointer is invalid.
[0]	1	The possible values of this bit are: `0` Entry is valid. `1` Entry is invalid.
Word 5		
[31:4]	28	Physical address.
[3:2]	2	The encodings of this field are: `2b00` Word which is not first or last of the TLB entry. `2b01` First word for that TLB entry. `2b10` Last word for that TLB entry. `2b11` First word for that TLB.
[1]	1	RESERVED.
[0]	1	The possible values of this bit are: `0` Entry is valid. `1` Entry is invalid.

3.4.5 Secure Alias to Non-secure Configuration Register 0

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

3.4.6 Secure Alias to Non-secure Auxiliary Configuration Register

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

3.4.7 Stream Match registers

The Stream Match registers characteristics are:

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

The MMU-400 supports Stream ID 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 stream ID. The Stream ID uniquely identifies the originator of a transaction, and you might commonly derive the Stream ID from identifier information conveyed on the bus interconnect:

- NS
- RnW
- ID.

You can identify a number of stream ID values as belonging to the same context. This permits the sharing of the state describing that context. Mapping multiple Stream ID 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 Stream ID, that is, a stream that has transactions in progress or that has issuing transactions, can match at most one entry in the Stream Match Register table. If the Stream ID 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 a Stream Match Register 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 Stream ID 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 are configured as described in *Configurable options* on page 1-5. 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 1-5. The unimplemented registers or those reserved by secure software and so not visible to non-secure access are RAZ/WI.

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

Figure 3-9 shows the Stream Match registers bit assignments.

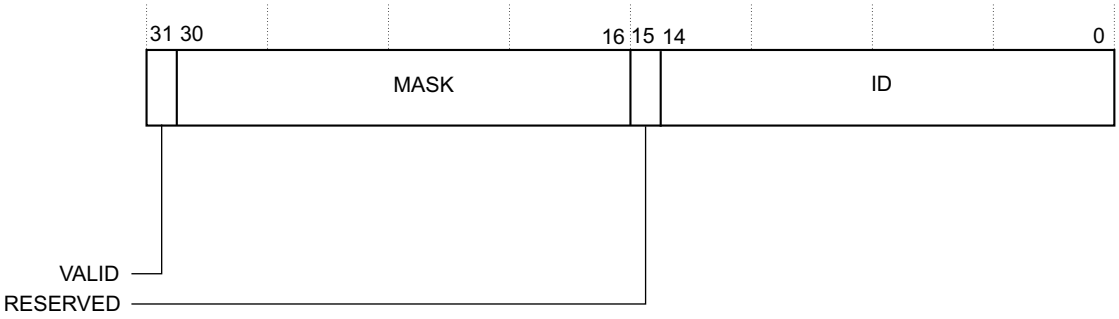


Figure 3-9 Stream Match registers bit assignments

Table 3-16 shows the Stream Match registers bit assignments.

Table 3-16 Stream Match registers bit assignments

Bits	Name	Description	
[31]	VALID	1	Entry included in Stream Mapping table search.
		0	Entry not included in Stream Mapping table search.
[30:16]	MASK	**MASK[i]**==1	ID[i] is ignored.
		MASK[i]==0	ID[i] is relevant to match.
[15]	RESERVED	RESERVED.	
[14:0]	ID	Stream ID to match.	

3.4.8 Stream to Context registers

The Stream to Context registers characteristics 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 1-5. 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* for more information.

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

3.5 The MMU-400 Global Register Space 1

The MMU-400 Global register space 1 gives 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 Context Bank Fault Restricted Syndrome Register A characteristics are:

Purpose The Context Bank Fault Restricted Syndrome registers A, SMMU_CBFRSYNRA*n*, hold 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 implementation supporting the 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.

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

Figure 3-10 shows the Context Bank Restricted Syndrome Register A bit assignments.

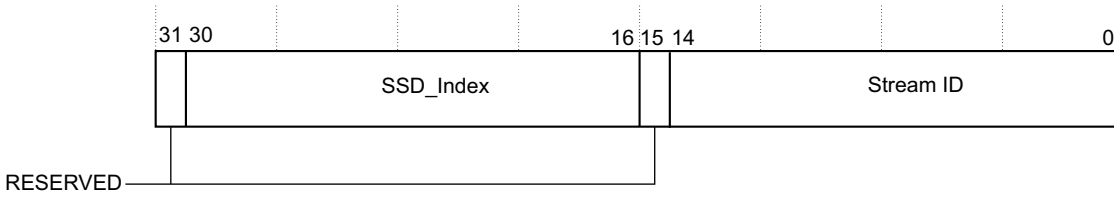


Figure 3-10 Context Bank Restricted Syndrome Register bit assignments

Table 3-17 shows the Context Bank Restricted Syndrome Register A bit assignments

Table 3-17 Context Bank Restricted Syndrome Register bit assignments

Bits	Name	Description
[31]	RESERVED	RESERVED.

Table 3-17 Context Bank Restricted Syndrome Register bit assignments (continued)

Bits	Name	Description
[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]	Stream ID	The Stream ID of the transaction that caused the fault.

3.5.2 Context Bank Attribute Register

The Context Bank Attribute Register characteristics are:

Purpose The Context Bank Attribute registers, SMMU_CBAR*n*, specify 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 3-18 shows.

Table 3-18 Context Bank Attribute Register

SMMU_CBARn[TYPE]	SMMU_CBARn Format	Description
00	Stage 2 Context	*Stage 2 Context, TYPE==00*
01	RESERVED	RESERVED
10	RESERVED	RESERVED
11	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 00, and format selected accordingly.

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

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

Stage 2 Context, TYPE==00

Figure 3-11 on page 3-32 shows the Stage 2 Context, TYPE==00 format that configures the associated Translation Context Bank to provide stage 2 translation.

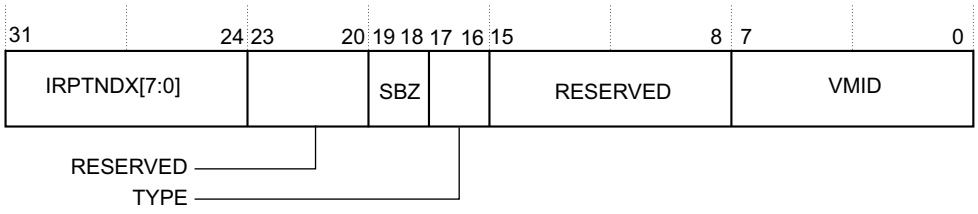


Figure 3-11 Stage 2 Context, TYPE==00 Register bit assignments

Table 3-19 shows the Stage 2 Context, TYPE==00 format that configures the associated Translation Context Bank to provide Stage 2 translation.

Table 3-19 Stage 2 Context, TYPE==00 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 specified context, that is, specified in an SMMU_S2CR*n* register.

3.6 Integration registers

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

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

3.6.1 Integration Mode Control Register

The ITCTRL Register characteristics are:

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

Note

When a device has been in integration mode, it might not function with the original behavior. After performing integration, you must reset the system to ensure 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.

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

Figure 3-13 on page 3-34 shows the bit assignments for the ITCTRL Register.

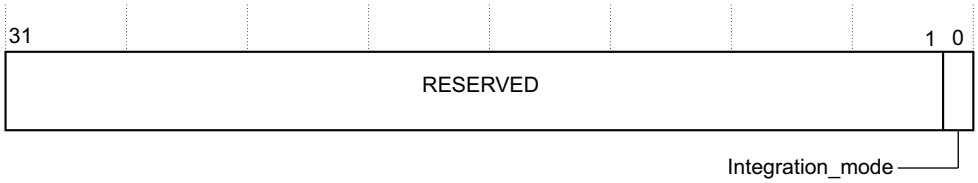


Figure 3-12 ITCTRL Register bit assignments

Table 3-21 on page 3-34 shows the ITCTRL Register bit assignment.

Table 3-20 ITCTRL Register bit assignments

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

3.6.2 Integration Test Input Register

The ITIP Register characteristics are:

Purpose Enables the status of the MMU-400 to be read.

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

Figure 3-13 shows the bit assignments for ITIP Register.

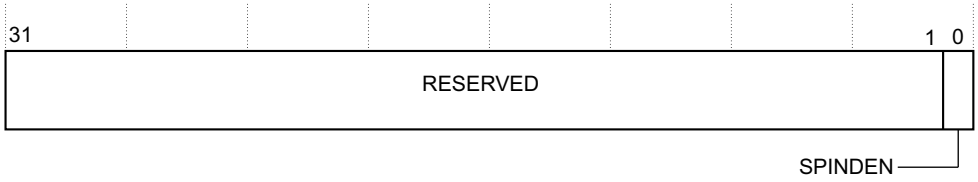


Figure 3-13 ITIP Register bit assignments

Table 3-21 shows the ITIP Register bit assignment.

Table 3-21 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 ITOP Register characteristics are:

Purpose Enable the status of the MMU-400 to be set in integration test mode. In integration mode, this register controls the outputs as Table 3-22 shows. It can be set in integration mode, by programming the integration control register as Table 3-20 on page 3-33 shows.

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

Figure 3-14 shows the bit assignments for ITOP Register.

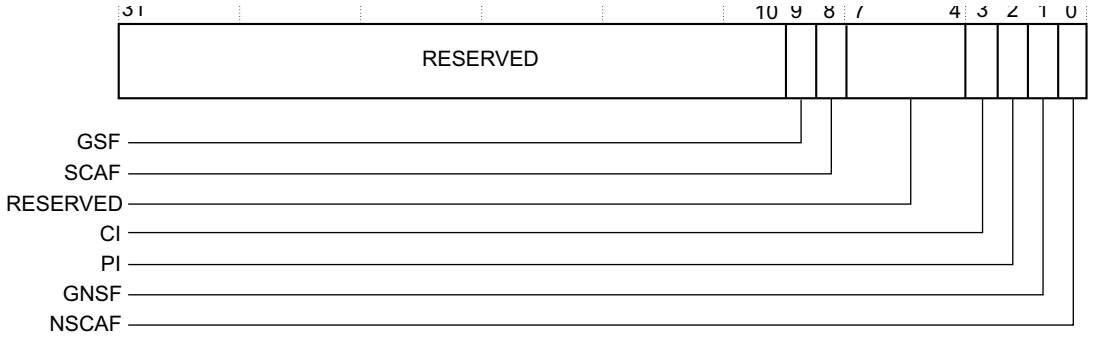


Figure 3-14 ITOP Register bit assignments

Table 3-22 shows the ITOP Register bit assignments

Table 3-22 ITOP Register bit assignments

Bits	Name	Description
[31:10]	RESERVED	RESERVED.
[9]	GSF	Global Secure Fault.
[8]	SCAF	Secure Configuration Access Fault.

Table 3-22 ITOP Register bit assignments (continued)

Bits	Name	Description
[7:4]	RESERVED	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 Counter Enable Set and Clear registers* on page 3-39
- *Performance Monitor Interrupt Enable Set and Clear registers* on page 3-40
- *Performance Monitor Overflow Status Set and Clear registers* on page 3-41
- *Performance Monitor Configuration Register* on page 3-42
- *Performance Monitor Control Register* on page 3-43
- *Performance Monitor Authentication Status Register* on page 3-45
- *Performance Monitor Device Type Register* on page 3-46.

3.7.1 Performance Monitor Event Count Register

The Performance Monitor Event Count Register characteristics are:

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

The size of the register, SIZE, is defined by PPMCFGR.SIZE field. See *Performance Monitor Configuration Register* on page 3-42.

Reads from PMEVCNTR*n* to 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.

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

Figure 3-15 shows the PMCVENT*n* format.

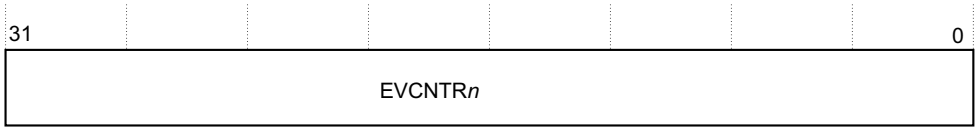


Figure 3-15 Performance Monitoring Event Count Register

Table 3-23 shows the PMCVENT*n* format.

Table 3-23 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 Performance Monitor Counter Group Configuration Register characteristics are:

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

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

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

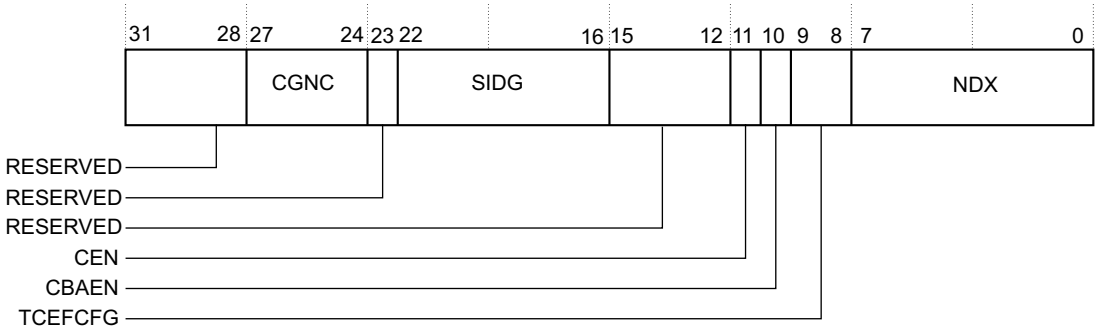


Figure 3-16 Performance Monitor Counter Group Configuration Register bit assignments

Table 3-24 shows the Performance Monitor Counter Group Configuration Register bit assignments.

Table 3-24 Performance Monitor Counter Group Configuration Register bit assignments

Bits	Name	Description
[31:28]	RESERVED	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 3.
[23]	RESERVED	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.
[15:12]	RESERVED	RESERVED.
[11]	CEN	Count Enable. Corresponds to the Performance Monitor Count Enable, PMCR.CEN, for this counter group.

Table 3-24 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!=10 or 01 then UNPREDICTABLE.
[9:8]	TCEFCFG	Translation Context Event Filtering Configuration. 00 Count Events on a global basis. 01 Count Events restricted to match in corresponding PMCGSMR*n*. 10 Count Events restricted to Translation Context Bank indicated by NDX. 11 RESERVED.
[7:0]	NDX	Index. Interpreted based on TCEFCFG, only valid if TCEFCFG==10 else RESERVED.

3.7.3 Performance Monitor Counter Group Stream Match Register

The Performance Monitor Counter Group Stream Match Register characteristics are:

Purpose The Performance Monitor Counter Group Stream Match registers, PMCGSMRn, specify the Stream ID-based filtering of events counted in a counter group.

The number of ID and MASK bits actually present is configured for Stream_ID_Width as *Configurable options* on page 1-5 describes. The unimplemented bits behave as RAZ/WI. An implementation provides the same number of ID and MASK bits for every implemented PMCGSMRn.

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

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

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

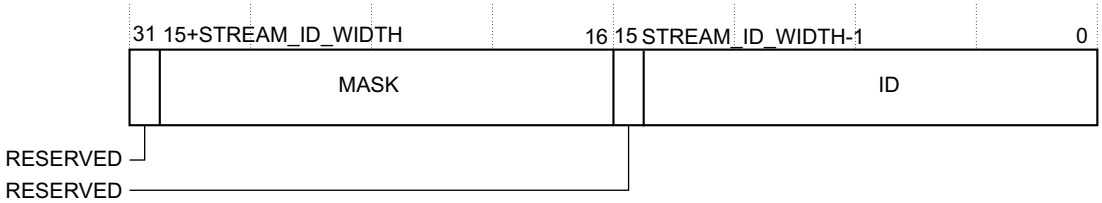


Figure 3-17 Performance Monitor Counter Group Stream Match Register bit assignments

Table 3-25 shows the Performance Monitor Counter Group Stream Match Register bit assignments.

Table 3-25 Performance Monitor Counter Group Stream Match Register bit assignments

Bits	Name	Description
[31]	RESERVED	RESERVED.

Table 3-25 Performance Monitor Counter Group Stream Match Register bit assignments

Bits	Name	Description	
[15+STREAM_ID_WIDTH:16]	MASK	**MASK[i]**==1 **MASK[i]**==0	ID[i] is ignored. ID[i] is relevant to match.
[15:STREAM_ID_WIDTH]	RESERVED	RESERVED.	
[STREAM_ID_WIDTH-1:0]	ID	Stream ID to match.	

3.7.4 Performance Monitor Counter Enable Set and Clear registers

The Performance Monitor Counter Enable Set and Clear registers characteristics are:

Purpose The Performance Monitor Counter Enable Set Register, PMCNTENSET is used to set the bits from the Counter Enable Register, CNTENR.

The Performance Monitor Counter Enable Clear Register, PMCNTENCLR is used to clear the bits, to read the value of the Counter Enable Register, CNTENR.

CNTENR is a 3-bit register. See *Performance Monitor Control Register* on page 3-43.

CNTENR is UNKNOWN on reset.

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

Figure 3-18 shows the Performance Monitor Counter Enable Set and Clear registers bit assignments.

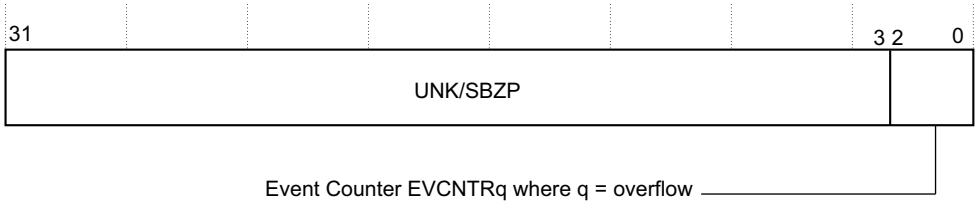


Figure 3-18 Performance Monitor Counter Enable Set and Clear registers bit assignments

Table 3-26 shows the Performance Monitor Counter Enable Set and Clear registers bit assignments.

Table 3-26 Performance Monitor Counter Enable Set and Clear registers bit assignments

Bits	Name	Reset value	Description
[31:3]	RESERVED	-	UNK/SBZP.
[2:0]	Event Counter EVCNTRq where q= overflow	0	Counter enable bit CNTENR[q], q refers to Event counter register EVCNTRq. On reads, each counter enable bit returns the corresponding bit of CNTENR, whether read through PMCNTENSET, or PMCNTENCLR. The action on writes depends on the register written through.

Table 3-27 shows the read and write bit values for the PMCNTEN0 registers.

Table 3-27 Read and write bit values for the PMCNTENn registers

Value	Meaning on reads	Register	Description
0	Counter is disabled	-	No action, write is ignored.
1	Counter is enabled	PMCNTENSET0	Enable counter. Set CNTENR bit to one.
		PMCNTENCLR0	Disable counter. Set CNTENR bit to zero.

3.7.5 Performance Monitor Interrupt Enable Set and Clear registers

The Performance Monitor Interrupt Enable Set and Clear registers characteristics are:

Purpose The Performance Monitor Interrupt Enable Set, PMINTENSET is used to set bits from the Interrupt Enable Register, INTENR.

The Performance Monitor Interrupt Enable Clear Register, PMINTENCLR, is used to clear the bits, to read the value of the Interrupt Enable Register, INTENR.

INTENR is a 3-bit register. See *Performance Monitor Control Register* on page 3-43.

Each bit, **INTENR[q]** controls the overflow interrupt for EVCNTq.

For more information, see *Performance Monitor Counter Enable Set and Clear registers* on page 3-39.

INTENR is UNKNOWN on reset. To avoid spurious interrupts being generated, software must set the interrupt enable values before enabling any of the counters. Interrupts are not signaled if the enable bit, PMCR.E, is set to 0.

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

Figure 3-19 shows the Performance Monitor Interrupt Enable Set and Clear registers bit assignments.

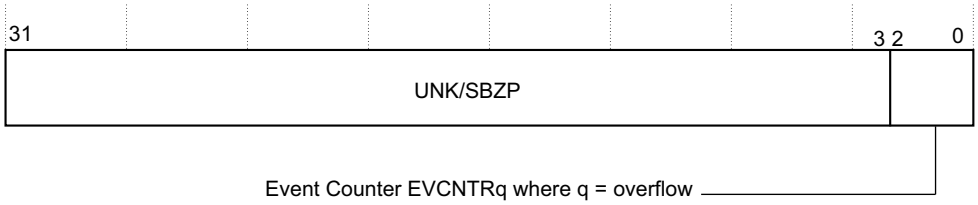


Figure 3-19 Performance Monitor Interrupt Enable Set and Clear registers bit assignments

Table 3-28 shows the Performance Monitor Interrupt Enable Set and Clear registers bit assignments.

Table 3-28 Performance Monitor Interrupt Enable Set and Clear registers bit assignments

Bits	Name	Description
[31:3]	RESERVED	UNK/SBZP.
[2:0]	Event Counter EVCNTRq where q = overflow	Interrupt enable bit INTENR[q], q refers to event counter register EVCNTRq. On reads, each interrupt enable bit returns the corresponding bit of INTENR, whether read through PMINTENSET, or PMINTENCLR. The action on writes depends on the register written through.

Table 3-27 on page 3-40 shows the read and write bit values for the PMCNTEN0 registers.

Table 3-29 Read and write bit values for the PMINTEN0 registers

Value	Meaning on reads	Register	Description
0	Overflow interrupt disabled	-	No action, write is ignored
1	Overflow interrupt enabled	PMINTENSET0	Enable overflow interrupt. Set INTENR bit to one.
		PMINTENCLR0	Disable overflow interrupt. Set INTENR bit to zero.

3.7.6 Performance Monitor Overflow Status Set and Clear registers

The Performance Monitor Overflow Status Set and Clear registers characteristics are:

Purpose The Performance Monitor Overflow Status Set and Clear registers, PMOVSSET0 and PMOVSCLR0, provide overflow status for the event counter registers.

PMOVSSET Sets the state of the overflow bit for each of the implemented event counters.

PMOVSCLR Clears the overflow status of the event counters

The overflow status is reset to 0.

Each PMOVSSET0 and PMOVSCLR0 registers contain the overflow status corresponding to 3 event counters, so PMOVSSET0 and PMOVSCLR0 correspond to event counters 0-2.

Each bit in a PMOVSSET0 and a PMOVSCLR0 register that correspond to unimplemented event counters are RESERVED.

A read of a PMOVSSET0 and PMOVSCLR0 registers return the current overflow status of the event counters that the register corresponds to.

A write to a PMOVSSET0 register has the following behavior:

`1` Bit causes the corresponding overflow status to be set.

`0` Bit is ignored.

A write to a PMOVSCLR0 register has the following behavior:

`1` Bit causes the corresponding overflow status to be cleared.

`0` Bit is ignored.

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

Figure 3-19 on page 3-40 shows the Performance Monitor Overflow Flag Set and Clear registers bit assignments.

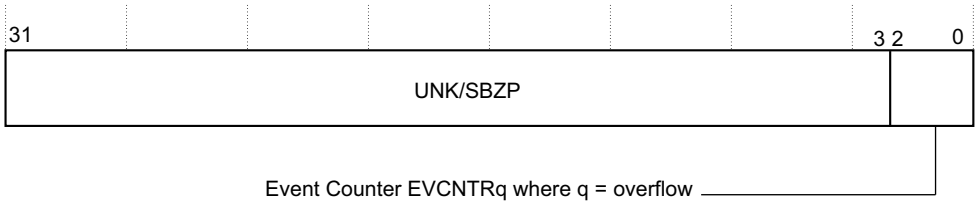


Figure 3-20 Performance Monitor Overflow Flag Set and Clear registers bit assignments

Table 3-28 on page 3-40 shows the Performance Monitor Overflow Flag Set and Clear registers bit assignments.

Table 3-30 Performance Monitor Overflow Flag Set and Clear registers bit assignments

Bits	Name	Description
[31:3]	RESERVED	UNK/SBZP.
[2:0]	Event Counter EVCNTRq where q = overflow	Overflow enable bit OVSR[q], q refers to event counter register EVCNTRq. On reads, each overflow flag status set or clear bit returns the corresponding bit of OVSR, whether read through PMOVSRSET, or PMOVSRCLR. The action on writes depends on the register written through.

3.7.7 Performance Monitor Configuration Register

The Performance Monitor Configuration Register characteristics are:

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

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

Figure 3-21 shows the Performance Monitor Configuration Register bit assignments.

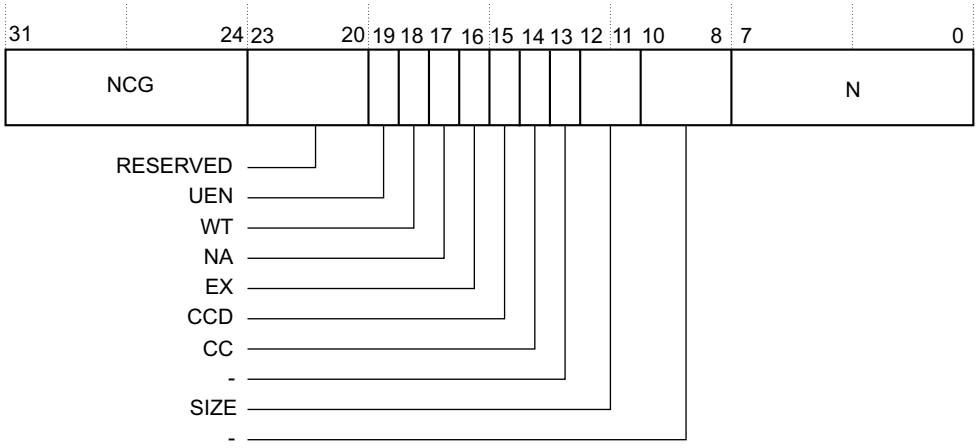


Figure 3-21 Performance Monitor Configuration Register bit assignments

Table 3-31 shows the Performance Monitor Configuration Register bit assignments.

Table 3-31 Performance Monitor Configuration Register bit assignments

Bits	Name	Reset value	Description
[31:24]	NCG	1	Number of counter groups.
[23:20]	RESERVED	-	RESERVED.
[19]	UEN	0	Reads as 0, user enable not supported.
[18]	WT	0	Reads as 0, two-state control state machine is implemented.
[17]	NA	0	Reads as 0, you can read the event counters at any time.

Table 3-31 Performance Monitor Configuration Register bit assignments (continued)

Bits	Name	Reset value	Description
[16]	EX	1	Reads as 1, export is supported. PMCR.EX is writable.
[15]	CCD	0	Reads as 0, no cycle counter pre-scale.
[14]	CC	0	Reads as 0, cycle counter not implemented
[13]	RESERVED	0	Reads as 0.
[12:11]	SIZE	11	Reads as 11, 32 bit event counters.
[10:8]	RESERVED	111	Reads as 111.
[7:0]	N	3	Indicates the number of implemented event counters.

3.7.8 Performance Monitor Control Register

The Performance Monitor Control Register characteristics 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.

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

Figure 3-22 shows the PMCR format.

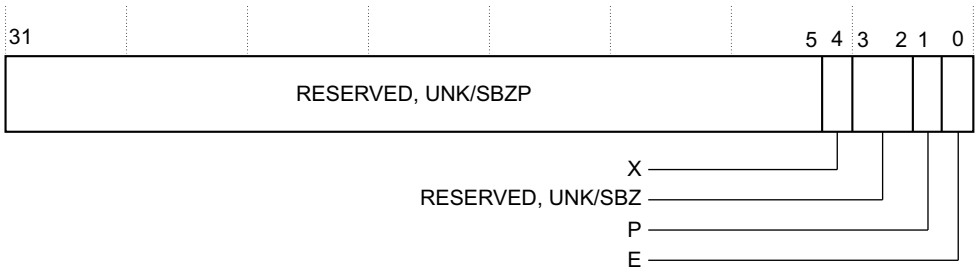


Figure 3-22 Performance Monitor Control Register format

Table 3-32 shows the PMCR format.

Table 3-32 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, like *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 3-33.

Table 3-33 shows the action on writes to the count enable bit

Table 3-33 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

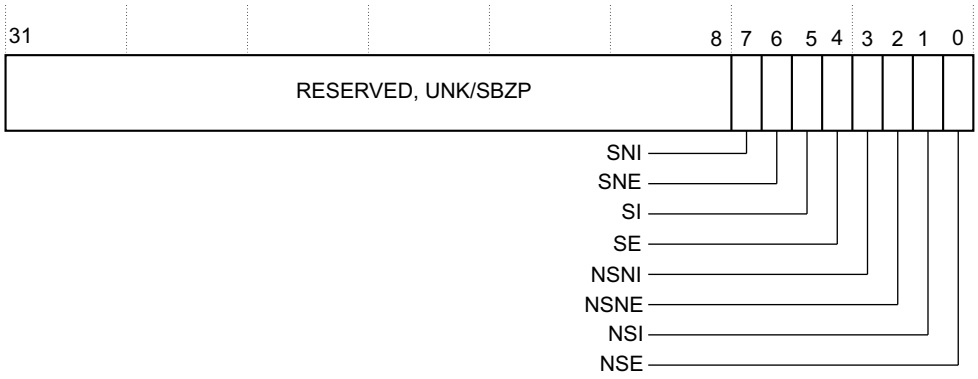
3.7.9 Performance Monitor Authentication Status Register

The Performance Monitor Authentication Status Register characteristics are:

Purpose The Performance Monitor Authentication Status Register, 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.

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

Figure 3-23 shows the Performance Monitor Authentication Status register bit assignments.



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

Figure 3-23 Performance Monitor Authentication Status Register format

Table 3-34 shows the Performance Monitor Authentication Status register bit assignments.

Table 3-34 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.

3.7.10 Performance Monitor Device Type Register

The Performance Monitor Device Type Register characteristics 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.

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

Figure 3-24 shows the Performance Monitor Device Type Register bit assignments.

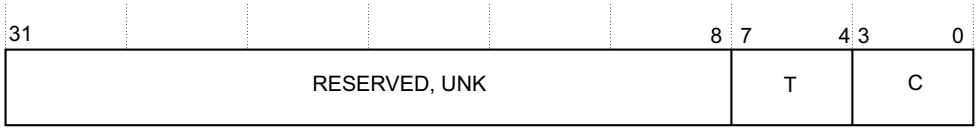


Figure 3-24 Performance Monitor Device Type Register bit assignments

Table 3-35 shows the Performance Monitor Device Type Register bit assignments.

Table 3-35 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 is `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.

Whether the security state determination registers are implemented that is described in *Configurable options* on page 1-5. 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 implementations where the registers are not present, this address space behaves as UNK/SBOP.

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

Table 3-36 shows the security state determination address space layout in terms of offsets from SMMU_GSSD_BASE.

Table 3-36 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 Component Identification registers, CID characteristics are:

Purpose The CID 0-3 registers, only bits [7:0] of each registers holds preamble information, and bits [31:8] are RESERVED.

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

Figure 3-25 shows the bit assignments for CID registers 0-3.

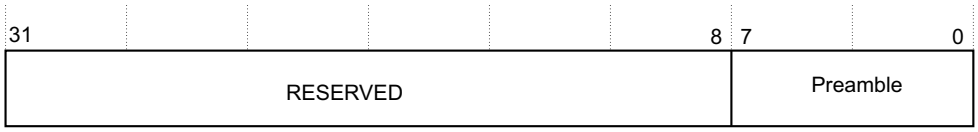


Figure 3-25 CID registers 0-3 bit assignments

Table 3-37 shows the register bit assignments for CID register 0-3

Table 3-37 CID Register 0-3 register bit assignments

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

3.9.2 Peripheral Identification registers

The Peripheral Identification registers, PID characteristics are:

Purpose In PID, only bits [7:0] of each register are used, and the Peripheral ID registers 7-5 are RESERVED.

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

The following are the Peripheral Identification registers:

- *Peripheral Identification Register 0*
- *Peripheral Identification Register 1* on page 3-49
- *Peripheral Identification Register 2* on page 3-49
- *Peripheral Identification Register 3* on page 3-50
- *Peripheral Identification Register 4* on page 3-50
- *Peripheral Identification registers 5-7* on page 3-51.

Peripheral Identification Register 0

Figure 3-26 on page 3-49 shows the register bit assignments for PID Register 0.

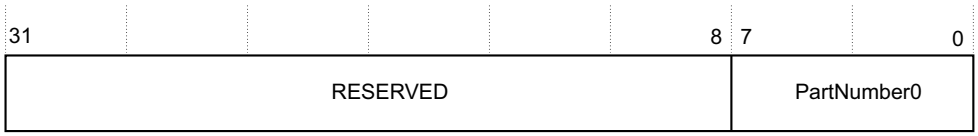


Figure 3-26 PID Register 0 register bit assignments

Table 3-38 shows the register bit assignments for PID Register 0.

Table 3-38 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 3-27 shows the register bit assignments for PID Register 1.

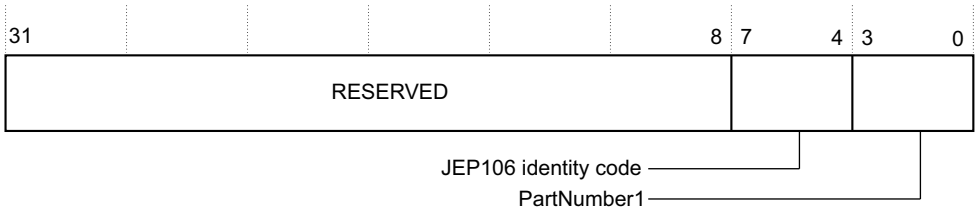


Figure 3-27 PID Register 1 register bit assignments

Table 3-39 shows the register bit assignments for PID Register 1.

Table 3-39 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 3-28 shows the register bit assignments for PID Register 2.

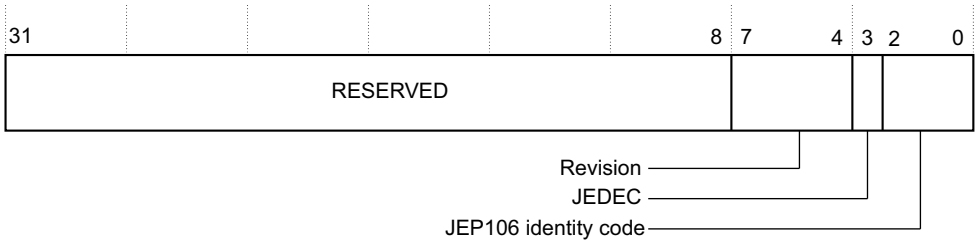


Figure 3-28 PID Register 2 register bit assignments

Table 3-40 shows the register bit assignments for PID Register 2

Table 3-40 PID Register 2 register bit assignments

Bits	Name	Reset value	Description
[31:8]	RESERVED	-	RESERVED.
[7:4]	Revision	0	Revision number of Peripheral. It starts from `0x0`.
[3]	JEDEC	1	Always set, indicates that a JEDEC assigned value is used.
[2:0]	JEP106 identity code	011	JEP106 identity code [6:4].

Peripheral Identification Register 3

Figure 3-29 shows the register bit assignments for PID Register 3.

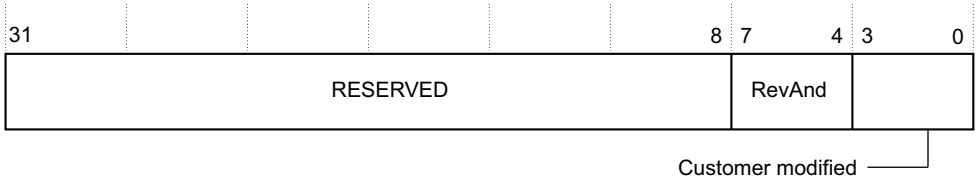


Figure 3-29 PID Register 3 register bit assignments

Table 3-41 shows the register bit assignments for PID Register 3

Table 3-41 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 3-30 shows the register bit assignments for PID Register 4.

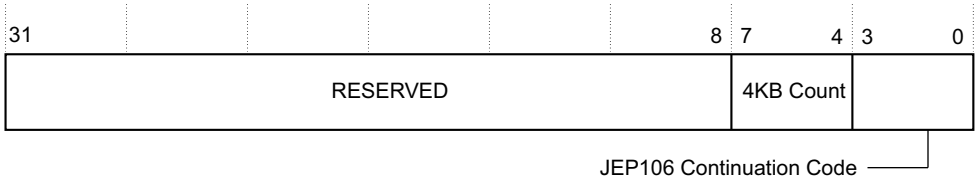


Figure 3-30 PID Register 4 register bit assignments

Table 3-42 shows the register bit assignments for PID Register 4

Table 3-42 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 3-31 shows the bit assignments for PID Registers 5-7.

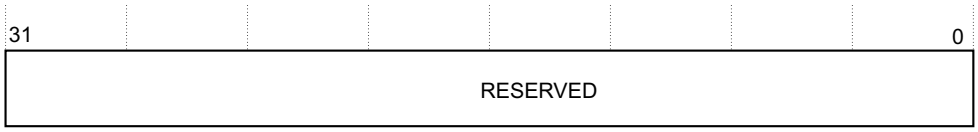


Figure 3-31 PID register 5-7 bit assignments

Table 3-43 shows the register bit assignments for PID Register 5-7

Table 3-43 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 System Control Register characteristics are:

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

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

Figure 3-32 shows the System Control Register bit assignment.

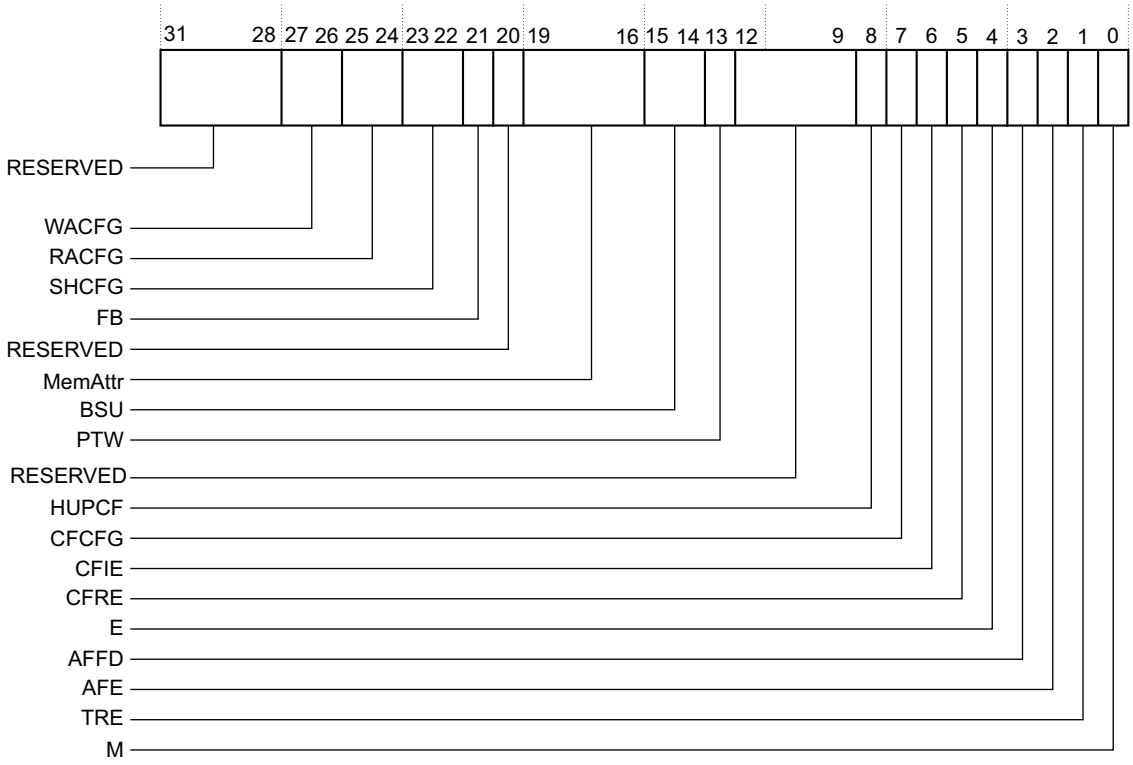


Figure 3-32 System Control Register bit assignment

Table 3-44 shows the System Control Register bit assignment.

Table 3-44 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: 00 Use the default allocation attributes. 01 RESERVED. 10 Write-Allocate. 11 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: 00 Use the default allocation attributes. 01 RESERVED. 10 Read-Allocate. 11 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: 00 Use shareable attribute as presented with transaction. 01 Outer Shareable. 10 Inner Shareable. 11 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 are permitted to be overlaid if SMMU_CB*n*_SCTLR.M has the value 0. Table 3-45 on page 3-55 and Table 3-46 on page 3-55 show valid values for this field

Table 3-44 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: `00` No effect. `01` Inner Shareable. `10` Outer Shareable. `11` 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 3-45 shows MemAttr bit values

Table 3-45 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 3-46 shows secondary MemAttr bit values

Table 3-46 Secondary MemAttr bit values

Bits[1:0]	Meaning when bits[3:2] == 00	Meaning when bits[3:2] != 00
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 Translation Table Base Control Register characteristics are:

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

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

Figure 3-33 shows the Translation Table Control Register bit assignments.

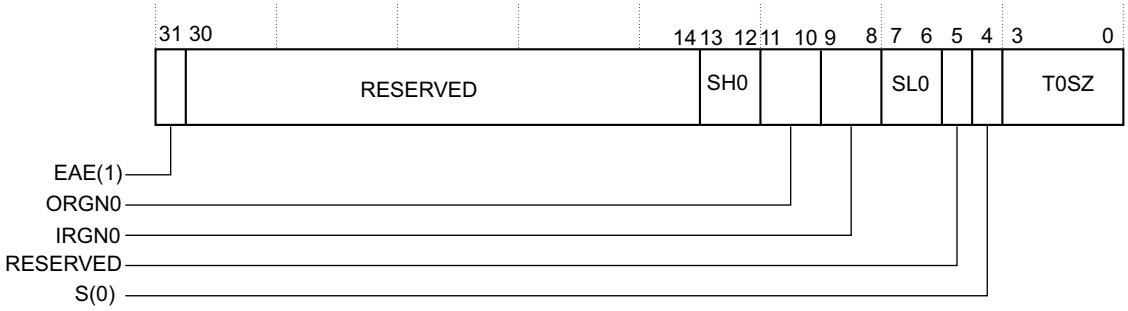


Figure 3-33 Translation Table Control Register bit assignments

Table 3-47 shows the Translation Table Control Register bit assignments.

Table 3-47 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	Sharebility 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]	SL0	Reserved.
[6]		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 Description

This appendix describes the AMBA and non-AMBA signals that the MMU-400 provides. It contain the following sections:

A.1 Clock and resets

This section describes the requirements for implementing the **CLK** and **RESETn** signals in 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 *AMBA™ AXI Protocol Specification* describes the AMBA AXI signals, ACE-Lite signals, and *Low-power Interface* (LPI) signals that the MMU-400 uses. This section describes:

- *AXI3 signals*
- *AXI4 signals* on page A-7
- *ACE-Lite signals* on page A-10
- *APB signals* on page A-14
- *LPI signals* on page A-15
- *Snoop channel signals* on page A-15.

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.

Table 1-2 on page 1-8 and Table 1-3 on page 1-8 describes the PTW width, master ID width, and slave ID width.

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	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 separate AXI configuration option is selected.

Also write address, write data, and write response signals of PTW are dummy signals that are connected internally to any logic.

Write data channel signals

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

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

AMBA equivalent	Master Port of PTW		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	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 of TLB block	Direction	Master port of PTW	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 of TLB block	Direction	Master port of PTW	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 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 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 PTW.

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

AMBA equivalent	Master port of PTW	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*
- *Write data channel signals* on page A-8
- *Write response channel signals* on page A-9
- *Read address channel signals* on page A-9
- *Read data channel signals* on page A-10.

Table 1-2 on page 1-8 and Table 1-3 on page 1-8 describes the PTW width, master ID width, and slave ID width.

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	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

Table A-12 Write address channel signals (continued)

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master Port of PTW	Direction
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 slave port of 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 master port of 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 PTW.

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

AMBA equivalent	Master Port of PTW		Direction
WDATA	**For 64-bit** **For 128-bit**	The data width is **wdata_s[63:0]** The data width is **wdata_s[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	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

Table A-17 Read address channel signals (continued)

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW	Direction
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-1:0]**	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.

Table A-18 Read data channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW	Direction
RID	**rid_s[I_S:0]**	Output	**rid_m[I_M:0]**	Input	**rid_ptw[I_P:0]**	Input
RDATA	**rdata_s[63:0]**	Output	**rdata_m[63:0]**	Input	**rdata_ptw[63:0]**	Input
RRESP	**rresp_s[1:0]**	Output	**rresp_m[1:0]**	Input	**rresp_ptw[3:0]**	Input
RLAST	**rlast_s[0]**	Output	**rlast_m[0]**	Input	**rlast_ptw[0]**	Input
RVALID	**rvalid_s[0]**	Output	**rvalid_m[0]**	Input	**rvalid_ptw[0]**	Input
RUSER	**ruser_s[RUSER_WIDTH-1:0]**	Output	**ruser_m[RUSER_WIDTH-1:0]**	Input	-	Input
RREADY	**rready_s[0]**	Input	**rready_m[0]**	Output	**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-11
- *Write data channel signals* on page A-11
- *Write response channel signals* on page A-12
- *Read address channel signals* on page A-13
- *Read data channel signals* on page A-13.

Table 1-2 on page 1-8 and Table 1-3 on page 1-8 describes the PTW width, master ID width, and slave ID width.

Write address channel signals

Table A-19 shows the ACE-Lite write address channel signals.

Table A-19 Write address channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW	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

Write data channel signals

Table A-20 shows the ACE-Lite write data channel signals for slave port of TLB block.

Table A-20 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
WUSER	**wuser_s[WUSER_WIDTH-1:0]**		Input
WREADY	**wready_s[0]**		Output

Table A-21 shows the ACE-Lite write data channel signals for master port of TLB block.

Table A-21 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-22 shows the ACE-Lite write data channel signals for master port of PTW block.

Table A-22 Write data channel signals-master port of PTW

AMBA equivalent	Master port of PTW		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-23 shows the ACE-Lite write response channel signals.

Table A-23 Write response channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW	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-24 shows the ACE-Lite read address channel signals.

Table A-24 Read address channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW	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-25 shows the ACE-Lite read data channel signals.

Table A-25 Read data channel signals

AMBA equivalent	Slave port of TLB block	Direction	Master port of TLB block	Direction	Master port of PTW	Direction
RID	**rid_s[I_S:0]**	Output	**rid_m[I_M:0]**	Input	**rid_ptw[I_P:0]**	Input
RDATA	**rdata_s[127:0]**	Output	**rdata_m[127:0]**	Input	**rdata_ptw[127:0]**	Input
RRESP[a]	**rresp_s[1:0]**	Output	**rresp_m[1:0]**	Input	**rresp_ptw[1:0]**	Input
RLAST	**rlast_s[0]**	Output	**rlast_m[0]**	Input	**rlast_ptw[0]**	Input
RVALID	**rvalid_s[0]**	Output	**rvalid_m[0]**	Input	**rvalid_ptw[0]**	Input
RUSER	**ruser_s[RUSER_WIDTH-1:0]**	Output	**ruser_m[RUSER_WIDTH-1:0]**	Input	-	-
RREADY	**rready_s[0]**	Input	**rready_m[0]**	Output	**rready_ptw[0]**	Output

a. If there is a shared ACE-Lite, then the width of **RRESP** is [3:0].

A.2.4 APB signals

This section describes the APB signals for the following:

- *APB4 signals*
- *APB3 signals*.

APB4 signals

Table A-26 shows the APB4 signals.

Table A-26 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-27 shows the APB3 secure and non-secure signals.

Table A-27 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 LPI signals

Table A-28 shows the LPI signals.

Table A-28 LPI signals

AMBA equivalent	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-29 shows the ACE-Lite snoop channel signals.

Table A-29 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				
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 Non-AMBA signals

This section describes the non-AMBA signals as follows:

- *Sideband signals*
- *Interrupt signals*
- *MBIST signals* on page A-17
- *Authentication interface signal* on page A-17
- *Tie-off signals* on page A-17
- *Performance event signals* on page A-18.

A.3.1 Sideband signals

Table A-30 shows the sideband signals.

Table A-30 Sideband signals

Signal	Type	Width	Description
rsb_ns	Input	1	Determines the non-secure state of an incoming read transaction. It can change along with **arvalid** assertion.
wsb_ns	Input	1	Determines the non-secure state of an incoming write transaction. It can change along with **awvalid** assertion.
wsb_ssd	Input	0-8	Sideband signal to indicate the SSD index. If **rsb_ns** or **wsb_ns** exist, then this signal does not exist. It can change along with **awvalid** assertion.
rsb_ssd	Input	0-8	Sideband signal to indicate the SSD index. If **rsb_ns** or **wsb_ns** exist, then this signal does not exist. It can change along with **arvalid** assertion
wsb_sid_s	Input	0-8	Sideband signal to indicate the Write Stream ID. It can change along with **awvalid** assertion.
rsb_sid_s	Input	0-8	Sideband signal to indicate the Read Stream ID. It can change along with **arvalid** assertion.

A.3.2 Interrupt signals

Table A-31 shows the interrupt signals generated from the MMU-400. See the *ARM System Memory Management Unit Architecture Specification* for more information.

Table A-31 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 multiplexor. ARM recommends no further insertion of MBIST multiplexors. This interface exists only when the RAM option is chosen for the TLB Data. See the *CoreLink MMU-400 System Memory Management Unit Implementation Guide* for more information.

Table A-32 shows the MBIST signals.

Table A-32 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	6	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.3.4 Authentication interface signal

The authentication interface disables AXI accesses. Table A-33 shows the authentication interface signal. See the *CoreSight Architecture Specification*.

Table A-33 Authentication Interface signal

Signal	Type	Width	Description
spniden	Input	1	Permits secure transfers to take place on the AXI master interface

A.3.5 Tie-off signals

Table A-34 shows the tie-off signals.

Table A-34 Tie-off signals

Signal	Type	Width	Description
cfg_cttw	Input	-	Tie-off to indicate whether the MMU-400 performs 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. The options are as follows: `0` Functional mode `1` Test mode

A.3.6 Performance event signals

Table A-35 shows the performance event signals.

Table A-35 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 theTLB 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 read 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 Issue A

Change	Location	Affects
No changes, first release	-	-