ARM® System Memory Management Unit

Architecture Specification

ARM®

ARM IHI 0062B (ID122412)

ARM System Memory Management Unit
Architecture Specification

Release Information

The following changes have been made to this document.

Change History

Date	Issue	Confidentiality	Change
23 March 2012	A	Confidential Beta	Issue A, Beta release of the specification
18 December 2012	B	Non-Confidential	Issue B, Final release of the specification

Proprietary Notice

ARM System MMU (SMMU) Architecture Specification Licence

This end user licence agreement ("licence") is a legal agreement between you (either a single individual, or single legal entity) and ARM Limited ("ARM") for the use of the relevant SMMU Architecture Specification accompanying this licence. ARM is only willing to license the relevant SMMU Architecture Specification to you on condition that you accept all of the terms in this licence. By clicking "I agree" or otherwise using or copying the relevant SMMU Architecture Specification you indicate that you agree to be bound by all the terms of this licence. If you do not agree to the terms of this licence, ARM is unwilling to license the relevant SMMU Architecture Specification to you and you may not use or copy the relevant SMMU Architecture Specification and you should promptly return the relevant SMMU Architecture Specification to ARM.

"LICENSEE" means You and your Subsidiaries.

"Subsidiary" means, if You are a single entity, any company the majority of whose voting shares is now or hereafter owned or controlled, directly or indirectly, by You. A company shall be a Subsidiary only for the period during which such control exists.

1. Subject to the provisions of Clauses 2, 3 and 4, ARM hereby grants to LICENSEE a perpetual, non-exclusive, non-transferable, royalty free, worldwide licence to:

(i) use and copy the relevant SMMU Architecture Specification for the purpose of developing and having developed products that comply with the relevant SMMU Architecture Specification;

(ii) manufacture and have manufactured products which either: (a) have been created by or for LICENSEE under the licence granted in Clause 1(i); or (b) incorporate a product(s) which has been created by a third party(s) under a licence granted by ARM in Clause 1(i) of such third party's ARM SMMU Architecture Specification Licence; and

(iii) offer to sell, sell, supply or otherwise distribute products which have either been (a) created by or for LICENSEE under the licence granted in Clause 1(i); or (b) manufactured by or for LICENSEE under the licence granted in Clause 1(ii).

2. LICENSEE hereby agrees that the licence granted in Clause 1 is subject to the following restrictions:

(i) where a product is created under Clause 1(i) or manufactured under Clause 1(ii) it must contain at least one processor core which has either been (a) developed by or for ARM; or (b) developed under licence from ARM;

(ii) the licences granted in Clause 1(iii) shall not extend to any portion or function of a product that is not itself compliant with part of the relevant SMMU Architecture Specification; and

(iii) no right is granted to LICENSEE to sublicense the rights granted to LICENSEE under this Agreement.

3. Except as specifically licensed in accordance with Clause 1, LICENSEE acquires no right, title or interest in any ARM technology or any intellectual property embodied therein. In no event shall the licences granted in accordance with Clause 1 be construed as granting LICENSEE, expressly or by implication, estoppel or otherwise, a licence to use any ARM technology except the relevant SMMU Architecture Specification.

4. **The relevant SMMU Architecture Specification is provided "as is" with no warranties express, implied or statutory, including but not limited to any warranty of satisfactory quality, merchantability, noninfringement or fitness for a particular purpose.**

5. No licence, express, implied or otherwise, is granted to LICENSEE, under the provisions of Clause 1, to use the ARM tradename in connection with the relevant SMMU Architecture Specification or any products based thereon. Nothing in Clause 1 shall be construed as authority for LICENSEE to make any representations on behalf of ARM in respect of the relevant SMMU Architecture Specification.

6. This Licence shall remain in force until terminated by you or by ARM. Without prejudice to any of its other rights if LICENSEE is in breach of any of the terms and conditions of this Licence then ARM may terminate this Licence immediately upon giving written notice to You. You may terminate this Licence at any time. Upon expiry or termination of this Licence by You or by ARM LICENSEE shall stop using the relevant SMMU Architecture Specification and destroy all copies of the relevant SMMU Architecture Specification in your possession together with all documentation and related materials. Upon expiry or termination of this Licence, the provisions of clauses 6 and 7 shall survive.

7. The validity, construction and performance of this Agreement shall be governed by English Law.

ARM contract references: **LES-PRE-20225 ARM System MMU (SMMU) Architecture Specification Licence**

ARM Limited, 110 Fulbourn Road, Cambridge, England CB1 9NJ.

Note

The term ARM is also used to refer to versions of the ARM architecture, for example ARMv7 refers to version 7 of the ARM architecture. The context makes it clear when the term is used in this way.

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.

Contents
ARM System Memory Management Unit Architecture Specification

Preface

This preface introduces the *ARM® System Memory Management Unit Architecture Specification*. It contains the following sections:

About this specification

This specification introduces the ARM System MMU architecture.

Intended audience

This specification is written for readers who are familiar with system memory management concepts, but who do not necessarily have any experience of the ARM architecture.

Using this specification

The information in this specification is organized into the following chapters:

Chapter 1 *Introduction*

Gives a brief introduction to the System MMU architecture.

Chapter 2 *The Translation Process*

Gives the steps performed by the System MMU when processing a transaction.

Chapter 3 *The Fault Model*

Gives fault conditions the System MMU might encounter, and describes how the System MMU handles these faults.

Chapter 4 *Address Translation Commands*

Describes address translation operations as software accessible commands that convert an input address to an output address using a specified translation. It includes a usage model and descriptions of fault handling, nested translation, and interaction with the Security Extensions.

Chapter 5 *Coherency Issues*

Describes the relationship between controlling software and the System MMU. It includes a description of the optional support for coherent translation table walks. It also describes maintenance operations.

Chapter 6 *Debug Support*

Describes the optional and implementation-dependent nature of debug support, with recommendation for providing TLB visibility. It also identifies a debug-specific security requirement associated with the Security Extensions.

Chapter 7 *System MMU Performance Monitors Extension*

Describes OPTIONAL support for performance monitoring functionality.

Chapter 8 *Security Extensions*

Describes OPTIONAL support of the Security Extensions.

Chapter 9 *System MMU Address Space*

Gives the System MMU register address map in terms of the System MMU address spaces.

Chapter 10 *System MMU Global Register Space 0*

Specifies the contents of Global Register Space 0. This register space is for high-level control of the System MMU resources. It is also used for mapping device transactions to the Translation context banks.

Chapter 11 *System MMU Global Register Space 1*

Specifies the contents of Global Register Space 1, which exists to accommodate the number of addresses in the global register space exceeding the capacity of a single memory page, when the page size is 4KB.

Chapter 12 *System MMU implementation defined Address Space*

Specifies an address space reserved for IMPLEMENTATION DEFINED purposes.

Chapter 13 *System MMU Performance Monitors Extension Register Map*

Gives the Performance Monitors register map for the System MMU architecture.

Chapter 14 *The Security State Determination Address Space*

Specifies the address space for the Security state determination part of the translation process.

Chapter 15 ***Stage 1 Translation Context Bank Format***

Specifies the Stage 1 Translation context bank format.

Chapter 16 ***Stage 2 Translation Context Bank Format***

Specifies the Stage 2 Translation context bank format.

Glossary Read this for definitions of some terms used in this specification.

Conventions

The following sections describe conventions that this book can use:

- *Typographic conventions*
- *Register names*
- *Numbers*
- *Pseudocode descriptions*.

Typographic conventions

The typographical conventions are:

italic	Introduces special terminology, and denotes citations.
bold	Denotes signal names, and is used for terms in descriptive lists, where appropriate.
`monospace`	Used for assembler syntax descriptions, pseudocode, and source code examples. Also used in the main text for instruction mnemonics and for references to other items appearing in assembler syntax descriptions, pseudocode, and source code examples.
SMALL CAPITALS	Used for a few terms that have specific technical meanings, and are included in the glossary.
Colored text	Indicates a link. This can be: • a URL, for example, `http://infocenter.arm.com` • a cross-reference, that includes the page number of the referenced information if it is not on the current page, for example, *Pseudocode descriptions* • a link, to a chapter or appendix, or to a glossary entry, or to the section of the document that defines the colored term, for example *Translation context bank*.

Register names

In a register name, *s* denotes the presence or absence of the Secure register prefix, S. For example, in an implementation that includes the Security Extensions, the register SMMU_*s*ACR is implemented as both:

- the Secure register SMMU_SACR
- the Non-secure register SMMU_ACR.

Numbers

Numbers are normally written in decimal. Binary numbers are preceded by `0b`, and hexadecimal numbers by `0x`. Both are written in a `monospace` font.

Pseudocode descriptions

This manual uses a form of pseudocode to provide precise descriptions of the specified functionality. This pseudocode is written in a `monospace` font.

Additional reading

This section lists relevant publications from ARM and third parties.

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

ARM publications

See the following documents for other information.

- *ARM® Architecture Reference Manual, ARMv7-A and ARMv7-R edition* (ARM DDI 0406)
- *CoreSight™ Architecture Specification* (ARM IHI 0029).

Feedback

ARM welcomes feedback on its documentation.

Feedback on this manual

If you have comments on the content of this manual, send e-mail to `errata@arm.com`. Give:

- the title
- the number, ARM IHI 0062B
- 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 contains a brief introduction to the ARM System *Memory Management Unit* (MMU) architecture. It contains the following section:

- *About the ARM System MMU architecture on page 1-18*.

1.1 About the ARM System MMU architecture

The ARM *System MMU* (SMMU) architecture provides a flexible implementation framework for a *Memory Management Unit* (MMU) implementation, with a number of IMPLEMENTATION DEFINED options.

The architecture can be used for a system-level MMU. It supports address translation from an *input address* to an *output address*, based on address mapping and memory attribute information held in *translation tables*.

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

The SMMU architecture also supports the concept of *translation regimes*, in which a required memory access might require two stages of address translation. For example, in virtualized processor implementation:

- An operating system defines the translation tables for its own memory accesses, and for accesses by applications running under it. It does this believing it is mapping the *virtual addresses* (VAs) used by the processor to *physical addresses* (PAs) in the physical memory system. However, it actually defines addresses in an *intermediate physical address* (IPA) memory map.
- A hypervisor defines the translation tables that translate the IPAs for a particular *guest operating system* to PAs.

This means that any memory access by a Guest OS, or by an application, requires two stages of translation, that together define a single translation regime:

- stage 1, from VA to IPA
- stage 2, from IPA to PA.

Within this system, the hypervisor must also define the required translation tables for its own memory accesses. These are in a separate translation regime, with only one stage of translation in which the stage 1 translation maps VAs to PAs.

A single stage of address translation can require multiple translation table lookups. In this case, each translation table lookup is described as a *level* of address lookup.

An implementation of the ARM System MMU architecture can provide:

- multiple transaction contexts, that apply to specific streams of transactions
- single or two stage translation
- for any stage of translation, multiple levels of address lookup, to provide fine-grained memory control
- fault handling, logging, and signaling functionality
- debug and OPTIONAL performance monitoring functionality.

Chapter 2
The Translation Process

This chapter describes the steps that the System MMU performs on receiving a memory access request. It contains the following sections:

Note

This specification uses a register name scheme described in *Register names on page xiii*. An understanding of this scheme is essential for the correct interpretation of register names.

2.1 Overview of address translation

At the memory system level, a System MMU controls:

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

Figure 2-1 on page 2-21 shows an overview of the address translation process.

An access to a System MMU is referred to as a *transaction*:

- a *client transaction* is an access by a client device, that the System MMU is to process
- a *configuration transaction* is an access by a device that accesses a register or a command in the System MMU configuration address space.

Security state determination identifies whether a transaction is from a Secure or Non-secure device.

Context determination identifies the *stage 1* or *stage 2 context* resources the System MMU uses to process a transaction. In some cases, the System MMU can be configured to permit a transaction to *bypass* the translation process, or to *fault* a transaction, regardless of the requested translation.

Stage 1 and stage 2 translation tables provide translation and memory attribute information. The transaction is then processed subject to required checks on the memory access.

The acceleration of translation through the use of TLB functionality is supported by the architecture. The System MMU architecture provides TLB maintenance operations to manage TLBs. The exact behavior of any TLB functionality is IMPLEMENTATION DEFINED.

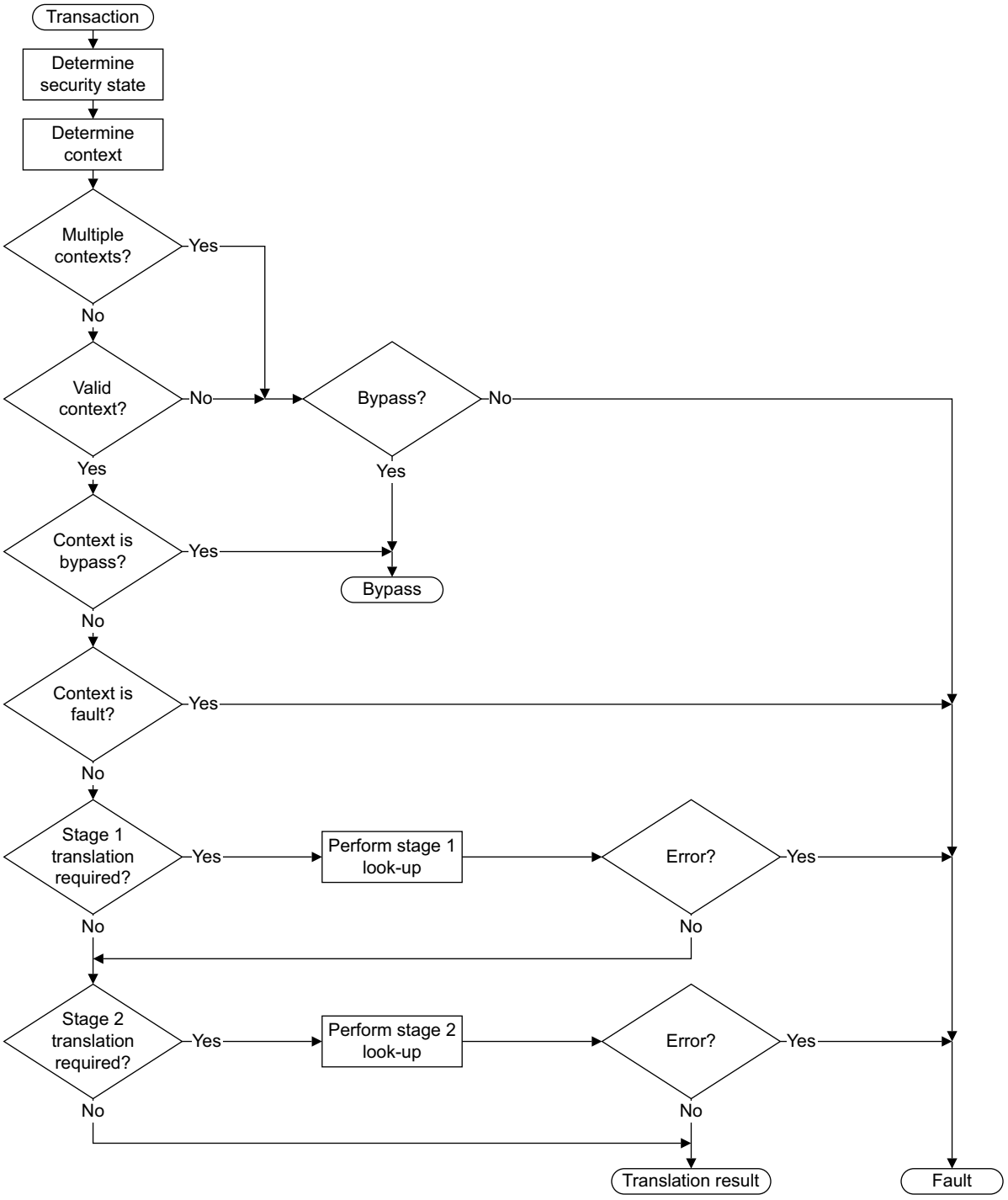


Figure 2-1 The address translation process

Note

For a transaction that requires two stages of address translation, as described in *About the ARM System MMU architecture on page 1-18*, translation table addresses for the stage 1 translation are typically defined in the IPA address space. If so, for each stage 1 lookup the System MMU must perform a stage 2 translation of the translation table address, to map the IPA to the corresponding PA. This stage 2 translation might fail, generating an error. Figure 2-1 does not show this possible dependence of a stage 1 lookup on a stage 2 translation.

2.2 Security state determination

The System MMU architecture supports shared use of a System MMU between Secure and Non-secure execution domains. The Secure execution domain can make both Secure and Non-secure MMU accesses. The Non-secure domain is limited to Non-secure accesses. The first step of transaction processing identifies which of these domains a transaction belongs to, and therefore the set of resources that process the transaction.

A device can transition between Secure and Normal ownership dynamically under the control of Secure software, and a Secure device can issue a transaction to Secure or Non-secure address space. Therefore, the security level, Secure or Non-secure, of a transaction arriving at the System MMU might be insufficient to determine whether a memory access originated from a Secure or a Non-secure device. In a system incorporating a System MMU, each transaction therefore has:

- a Secure or Non-secure memory access status
- a *transaction security state* attribute indicating whether the transaction originated from a Secure or Non-secure device.

The mechanism by which a system determines the security state of a transaction depends on whether the security state determination address space is present, as defined by the SMMU_IDR1 SSDTP bit. When this address space is present, SMMU_SSDR registers are provided in the System MMU address space.

Security state determination address is present

There are a number of ways in which a system might determine the security state of a transaction. The System MMU architecture does not define how this state is determined, but reserves space in the System MMU address space to provide an address-mapped bit vector that permits the security state to be determined from an SSD_Index associated with the transaction. See *System MMU security state determination address space on page 14-184* for more information.

Before transferring control to Non-secure software, Secure software must access the the appropriate SMMU_SSDRn register and:

1. set to 0 all bits corresponding to devices that must be Secure
2. set to 1 all other bits
3. read all bits to verify correct operation.

Security state determination address is not present

A System MMU implementation and the system that incorporates it can adopt alternative IMPLEMENTATION DEFINED approaches to determine the security state of a transaction. For example:

- the transaction source can determine a security state and propagate this with the transaction to the System MMU
- the System MMU can use the Secure memory access status in systems where all sources of Secure transactions only issue transactions with a Secure transaction status
- the System MMU can use other platform-specific knowledge to determine the security status of each transaction.

In either case, to prevent any device incorrectly being configured as Secure, Secure software must:

- be aware of the security state determination mechanism
- be aware of all devices that are permitted to be classed as Secure
- be able to identify the system to ensure the appropriate security state determination mechanism is used.

2.2.1 Banked registers

Some System MMU registers are Banked for security. A Non-secure access to a register address accesses the Non-secure copy of the register. A Secure access accesses the Secure copy in the Secure address space, at the same address offset as its Non-secure counterpart. For a configuration transaction, the security level determines which register is accessed. For a client transaction, the security state of the transaction determines which resource to use to process the transaction.

Not all registers available in both the Secure and Non-secure security states are Banked. The registers that are not Banked are:

- in translation contexts that are reserved by Secure or Non-secure software

- global address translation commands that are common to the Secure and Non-secure security states.

For information about Banked registers and naming conventions, see *Register names on page xiii*.

2.3 Context determination

The System MMU processes a transaction in one of the following ways:

- bypass translation altogether
- fault the transaction
- require the resources of one or more translation contexts, each having its own set of translations, attributes and permissions that apply to a transaction being processed by that context.

Context determination determines which resources the System MMU uses to process a transaction.

In a System MMU that supports two stages of translation, a transaction can be associated with up to two translation contexts.

2.3.1 Transaction streams

The System MMU can process transactions from multiple sources, potentially using a different translation context for each transaction. A *transaction stream* is a sequence of transactions associated with a particular thread of activity in the system.

Transactions from the same transaction stream are associated with the same translation context, and are therefore subject to the same type of processing in the System MMU. A device in the system can issue transactions using more than one transaction stream, and a single transaction stream can contain transactions from more than one device.

A *Stream Identifier* (StreamID) associates a transaction with a transaction stream. The StreamID is derived from transaction identification information carried with the transaction by the system interconnect. The StreamID uniquely identifies the originator of a transaction, and can commonly be derived from identifier information conveyed on the bus interconnect, such as the NS bit, the *Read not Write* (RnW) bit indicating whether a transaction is a read or a write operation, and the transaction ID.

The number of implemented StreamID bits:

- is defined by SMMU_IDR0.NUMSIDB[3:0]
- lies in the range 0-15.

A zero-bit StreamID might exist if the source of a single StreamID has a dedicated System MMU.

As a result of the system-specific nature of transaction identification, and the variety of system interconnect protocols that exist, the encoding of the StreamID used by a specific System MMU instance is IMPLEMENTATION DEFINED. For example, a StreamID might be derived from the following transaction identification information:

- the transaction ID
- the read and write status
- the security state.

2.4 The Stream mapping table

In the System MMU, the Stream mapping table maps each transaction to a transaction stream and its corresponding translation context. A System MMU implementation supports one of the following stream mapping schemes:

- StreamID matching
- StreamID indexing.

It is IMPLEMENTATION DEFINED whether a Stream mapping table implements the StreamID matching or StreamID indexing scheme. SMMU_IDR0.SMS identifies the implemented scheme.

The Stream mapping table consists of a number of entries. Each entry is a Stream mapping register group containing the following registers, where `n` defines the Stream mapping register group number:

- SMMU_SMRn
- SMMU_S2CRn.

In an implementation that uses StreamID matching, SMMU_SMRn determines whether a transaction matches the group. In an implementation that uses StreamID indexing, the StreamID is an index into the Stream mapping table.

SMMU_S2CRn specifies the first translation context, bypass attributes, or fault condition for the translation process.

SMMU_IDR0.NUMSMRG specifies the number of IMPLEMENTATION DEFINED Stream mapping register groups.

An attempt by Non-secure software to access a group above the reported implemented number results in one of the following IMPLEMENTATION DEFINED behaviors:

- the access is treated as RAZ/WI
- a Configuration access fault is raised, as Chapter 3 *The Fault Model* describes.

In an implementation that includes StreamID matching, ARM recommends that Secure software gives Non-secure software at least one Stream Match Register group.

2.4.1 No match handling

If no match for a transaction is found in the Stream mapping table, the System MMU can either:

- permit the transaction to bypass the System MMU translation process
- fault the transaction, resulting in an Unidentified Stream Fault.

SMMU_sCR0.USFCFG can be used to configure the handling of a transaction that does not match any Stream mapping table entry. In an implementation that includes the Security Extensions:

- SMMU_CR0.USFCFG only applies to Non-secure transactions
- SMMU_SCR0.USFCFG applies to Secure transactions.

Transactions that bypass the System MMU translation process are not subject to address translation. SMMU_sCR0 can be used to configure a set of bypass memory attributes that apply to all such transactions.

2.4.2 StreamID matching

In a register group that uses StreamID matching, SMMU_SMRn specifies conditions that must be met for a transaction to be associated with the Stream mapping register group to which SMMU_SMRn belongs.

The SMMU_SMRn registers form a table that is searched associatively to find a match for the StreamID of a transaction. For more information about StreamIDs, see *Transaction streams on page 2-24*.

If a transaction matches all of the conditions specified in SMMU_SMRn, the translation context, Bypass mode, or fault context specified by SMMU_S2CRn is used to process the transaction.

SMMU_SMRn provides StreamID and mask fields that permit the masking of StreamID bits irrelevant to the matching process.

The Stream mapping table permits a number of StreamID values to be mapped to the same translation context. This means the state describing that context can be shared. Mapping multiple StreamID values to the same translation context is achieved using multiple Stream mapping table entries, or using the mask facilities in the SMMU_SMRn encoding.

The Stream mapping table must be configured so that a StreamID matches, at most, one entry in the Stream mapping table. During configuration, software must ensure that there is no overlap in Stream mapping table entries for all StreamIDs that are active.

If the StreamID of a transaction matches multiple Stream mapping table entries, either of the following IMPLEMENTATION DEFINED options are possible:

- The System MMU detects the multiple match and either:
  - permits the transaction to bypass the System MMU translation process
  - faults the transaction with a Stream match conflict fault.
- The System MMU does not detect the multiple match, and processes the transaction using one of the matching entries in the Stream mapping table.

SMMU_IDR1.SMCD indicates whether the System MMU detects all Stream match conflicts. This value is IMPLEMENTATION DEFINED.

SMMU_sCR0.SMCFCFG specifies whether the System MMU permits bypass or raises a Stream match conflict fault. It is IMPLEMENTATION DEFINED whether this setting is configurable. If it is not configurable, SMMU_sCR0.SMCFCFG has a fixed value and writes are ignored.

SMMU_sCR0.USFCFG specifies how transactions with no match in the Stream mapping table are handled. It is IMPLEMENTATION DEFINED whether the System MMU supports all of the possible handling options.

Unimplemented SMMU_SMRn registers, or those reserved by Secure software and therefore invisible to Non-secure accesses, behave as RAZ/WI. These registers have an IMPLEMENTATION DEFINED option to trap accesses to unimplemented registers and raise a Configuration access fault. See Chapter 3 *The Fault Model*.

2.4.3 StreamID indexing

In an implementation that uses StreamID indexing, a one-to-one stream mapping associates a transaction with a Stream mapping register group. The StreamID associated with the transaction is an index into the Stream mapping table, and directly selects the Stream mapping register group for processing the transaction. The maximum StreamID size in a register group that uses StreamID indexing is 7 bits.

It is IMPLEMENTATION DEFINED whether the Stream mapping table implements Stream Match Register functionality. SMMU_IDR0.SMS indicates whether the register group supports Stream Match Register functionality. In an implementation that supports StreamID indexing, a Stream Match Register has no effect on the stream mapping process and is UNK/SBPZ.

2.4.4 Bypassing the Stream mapping table

A transaction bypasses the Stream mapping table if any of the following apply:

- SMMU_sCR0.CLIENTPD is 1
- the transaction does not match any entries in the Stream mapping table and SMMU_sCR0.USFCFG is 0
- the transaction is detected to match multiple entries in the Stream mapping table and SMMU_sCR0.SMCFCFG is 0.

A transaction that bypasses the Stream mapping table is not subject to address translation, and in most cases, is not subject to any protection checking:

- a data access is not subject to any protection checking
- an instruction fetch might be subject to a permission check if the implementation includes the Security Extensions.

Interaction with the Security Extensions

In an implementation that includes the Security Extensions:

- SMMU_CR0.CLIENTPD only applies to Non-secure transactions
- SMMU_SCR0.CLIENTPD is an equivalent field that applies to Secure transactions.

See *SMMU_sCR0, Configuration Register 0 on page 10-120*.

Secure instruction fetch transactions that bypass the Stream mapping table can be subject to a protection check. If the SMMU_SCR1.SIF bit is set to 1, a permission fault is recorded when a Secure domain access attempts to exit as a Non-secure instruction.

This check applies globally, and records faults to SMMU_SGFSR. If a transaction is associated with a particular Translation context bank, faults are recorded in SMMU_CBn_FSR instead of SMMU_SGFSR.

Instruction fetches by a Non-secure client are not subject to any protection checking.

Note

Whether Secure Instruction Fetch checks apply to instruction writes is IMPLEMENTATION DEFINED.

2.4.5 Stream-to-Context Register, SMMU_S2CR*n*

SMMU_S2CRn specifies the initial translation context for processing a transaction. *n* is in the range 0-127, and identifies the Stream mapping register group for the transaction. This means that the other register in the Stream mapping register group is SMMU_SMRn.

In an implementation that supports two stages of translation, the SMMU_CBARn register associated with the initial translation context can specify a subsequent translation context.

SMMU_S2CRn specifies one of the following initial translation options:

- a Translation context bank
  - a Stage 1 Translation context bank, if the implementation supports Stage 1 translation
  - a Stage 2 Translation context bank, if the implementation supports Stage 2 translation.
- Bypass mode, optionally overlaying memory attributes
- the fault context.

Translation context bank

An implementation can support the following types of translation:

- stage 1 only
- stage 2 only
- stage 1 followed by stage 2, described as *nested* translation.

If SMMU_S2CRn specifies a Stage 1 Translation context bank or a Stage 2 Translation context bank, the specified Translation context bank is the initial context for the translation. The SMMU_CBARn register associated with the Translation context bank can specify additional translation attributes.

In an implementation that supports nested translation, when a Stage 1 Translation context bank is specified for the initial translation, SMMU_CBARn can specify an associated Stage 2 Translation context bank.

The number of SMMU_CBARn registers matches the number of entries in the Translation context bank table, and is IMPLEMENTATION DEFINED. SMMU_IDR1.NUMCB specifies the number of implemented SMMU_CBARn registers.

For more information about attributes associated with a Translation context bank, see *The Context Bank Attribute Register, SMMU_CBARn on page 2-32*.

SMMU_S2CRn can specify the following memory attributes to replace those specified by the transaction:

- memory type, using the MemAttr field when MTCFG is set to 1
- shareable attributes, using the SHCFG field

- cache allocation hints, using the WACFG and RACFG fields
- instruction fetch attributes, using the INSTCFG field
- privilege attribute, using the PRIVCFG field
- for Secure Stream mapping register groups only, Non-secure configuration, using the NSCFG field.

These attributes are the starting point for translation. The Translation context bank can modify them. See *Memory type and shareability attributes on page 2-39* for more information.

It is IMPLEMENTATION DEFINED whether a System MMU implementation supports:

- Stage 1 translation. SMMU_IDR0.S1TS specifies whether Stage 1 translation is supported.
- Stage 2 translation. SMMU_IDR0.S2TS specifies whether Stage 2 translation is supported.
- Stage 1 and stage 2 nested translation. SMMU_IDR0.NTS specifies whether nested translation is supported.

Note

If nested translation is supported, then both stage 1 and stage 2 translation must be supported. That is, if the SMMU_IDR0.NTS bit is set to 1, then both the SMMU_IDR0.S1TS and SMMU_IDR0.S2TS bits must be set to 1.

The number of Translation context bank entries for stage 1 or stage 2 translation is IMPLEMENTATION DEFINED. If SMMU_S2CRn is configured to specify an unimplemented Translation context bank for stage 1 or stage 2, any transaction processed as part of that stream results in an Unimplemented context bank fault.

An SMMU_S2CRn register reserved by Secure software using SMMU_SCR1.NSNUMSMRGO must only specify a Translation context bank that is reserved by Secure software.

An SMMU_S2CRn register that is accessible from the Non-secure state must only specify a Translation context bank that is not reserved by Secure software.

If software does not comply with these restrictions, an Unimplemented context bank fault might arise when a transaction is mapped to an incorrectly configured Stream-to-Context Register.

Bypass mode

If SMMU_S2CRn.TYPE is configured as Bypass mode, no translation is applied to the transaction. However, SMMU_S2CRn can specify memory attributes for the transaction. For example, memory type, shareable attributes and cache allocation hints. See SMMU_S2CRn for more information.

No protection check is applied, except where an instruction fetch by a Secure client is subject to a protection check, based on the value of SMMU_SCR1.SIF.

Fault context

If SMMU_S2CRn specifies the fault context, all transactions associated with that Stream mapping register group are subject to an Invalid context fault. See *Global faults on page 3-56* for details of Invalid context faults.

2.4.6 Interaction with the Security Extensions

In an implementation that includes the Security Extensions, the Stream mapping table is a common resource. Secure software can reduce the size of the Stream mapping table seen by Non-secure software using SMMU_SCR1.NSNUMSMRGO. Non-secure accesses see only the table size defined by SMMU_SCR1.NSNUMSMRGO, and this value determines the size of the table seen by a Non-secure read of SMMU_IDR0.NUMSMRG. This functionality permits Secure software to reserve a number of Stream mapping register groups, for use by Secure software.

Secure software can access all implemented entries in the table. This means Secure software can inspect Non-secure Stream mapping table entries.

The stream mapping process uses the Secure status indication from the security state determination to prevent Non-secure access, as follows:

- transactions from Non-secure devices can only match Stream mapping table entries unreserved by Secure software

- transactions from Secure devices can only match Stream mapping table entries reserved by Secure software.

SMMU_S2CRn registers must specify a Translation context bank of appropriate security, as follows:

- An SMMU_S2CRn register reserved by Secure software can only specify a Translation context bank reserved by Secure software. If Secure software fails to comply with this requirement, the result is UNPREDICTABLE.
- An SMMU_S2CRn register not reserved by Secure software can only specify a Translation context bank that is not reserved by Secure software. If Non-secure software fails to comply with this requirement, an Unimplemented context bank fault is raised for any transactions mapped to that SMMU_S2CRn register.

2.4.7 Reset state

On reset, no initialization is performed on Stream mapping table entries. The required contents of the Stream mapping table must be initialized before the System MMU is enabled.

2.5 Translation context

A translation context provides information and resources required by the System MMU to process a transaction.

The System MMU can process multiple transaction streams from different threads of execution, and supports multiple live translation contexts.

A Translation context bank includes state for configuring the translation process and capturing fault status, and operations for maintaining cached translations. A Translation context bank specifies largely the same state used by the ARM processor architecture translation process, principally:

- the translation table base addresses
- memory attributes to use during the translation table walk
- translation table attribute remapping.

The format of a Translation context bank depends on whether it is used for stage 1 or stage 2 translation.

Context determination determines which Translation context bank is to be used during the processing of a transaction. Up to two translation contexts can be specified:

- on an implementation that does not support nested translations, only one translation context can be specified
- on an implementation that supports nested translations:
  - one translation context is specified for single-stage translation
  - two contexts are specified for a nested two-stage translation.

2.5.1 The Translation context bank table

Translation context banks are arranged as a table in the System MMU configuration address map. Each entry in the table occupies a *PAGESIZE* address space, where *PAGESIZE* is 4KB or 64KB. See *PAGESIZE and NUMPAGENDXB on page 9-98*.

The System MMU architecture provides space for up to 128 Translation context banks. The *PAGESIZE* alignment means a hypervisor can give a Guest OS direct access to a specific Translation context bank by using the stage 2 translation functionality provided by an MMU on the processor. Similarly, a hypervisor can use address translation to give the illusion of a contiguous Translation context bank table to a Guest OS.

A single Translation context bank table contains all of the Translation context banks, regardless of whether they are stage 1 or stage 2 format.

The number of Translation context banks a System MMU implements is IMPLEMENTATION DEFINED, and is specified by SMMU_IDR1.NUMCB. An attempt to access an unimplemented Translation context bank results in one of the following IMPLEMENTATION DEFINED behaviors:

- the access is treated as RAZ/WI.
- the access generates a Configuration access fault, as Chapter 3 *The Fault Model* describes.

Some Translation context banks might only support stage 2 translations.

SMMU_IDR1.NUMS2CB specifies the IMPLEMENTATION DEFINED number of Translation context banks that only support stage 2 translation. The remaining Translation context banks support either stage 1 or stage 2 translation. Translation context banks that only support stage 2 translation are grouped together, starting at location 0 in the Translation context bank table. The remaining Translation context banks are similarly grouped and start immediately after the Translation context banks that only support stage 2 translations.

For Translation context banks that can be used for either stage 1 or stage 2 translations, the translation format is specified by the SMMU_CBARn.TYPE field associated with the Translation context bank.

Interaction with the Security Extensions

In an implementation that includes the Security Extensions, SMMU_SCR1.NSNUMCBO can be used to reserve a number of Translation context banks for Secure use. This field reserves Translation context banks from the top of the Translation context bank table. Because only stage 1 translation is supported for Secure transactions, only Translation context banks that can support stage 1 translation can be reserved for Secure use.

For configuration accesses, Secure software can access all Translation context banks, and Non-secure software can only access Translation context banks that are not reserved for Secure use. This means Secure software can inspect Non-secure Translation context banks.

The value written to SMMU_SCR1.NSNUMCBO affects the value read from SMMU_IDR1.NUMCB.

Figure 2-2 gives an overview of the Translation context bank table, SMMU_IDR1 fields and the effect of the override registers.

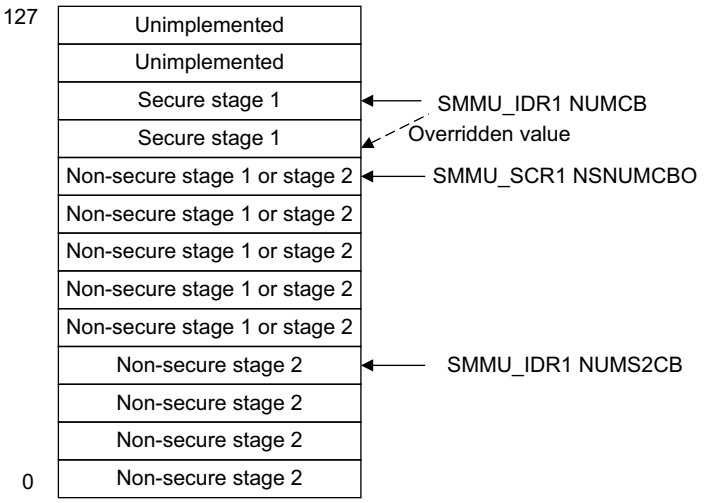


Figure 2-2 Translation context bank table

An SMMU_S2CRn register that is reserved for Secure use must specify a Translation context bank that is reserved by Secure software. An SMMU_S2CRn register that is not reserved for Secure use, or in the case of nested translation, an SMMU_CBARn register that is not reserved for Secure use, must specify a Translation context bank that is not reserved for Secure use. Otherwise, an Unimplemented context bank fault occurs when a transaction is mapped to the wrongly-programmed Translation context bank.

Translation context bank for stage 1 translation

A Translation context bank for stage 1 translation contains the following:

- context control registers
- translation table base and control registers
- fault address and status registers
- physical address and status registers
- TLB maintenance operations.

See Chapter 15 *Stage 1 Translation Context Bank Format* for more details.

Translation context bank for stage 2 translation

The format of a Translation context bank configured for stage 2 translation is similar to the stage 1 format. See *Stage 1 and Stage 2 context bank format differences on page 16-228* for information about the differences, and see Chapter 16 *Stage 2 Translation Context Bank Format* for more information about the stage 2 format.

Reset state

On reset, no initialization is performed on Translation context bank table entries. The Translation context bank table entries must be initialized to the required values before use.

2.5.2 The Context Bank Attribute Register, SMMU_CBAR*n*

Each Translation context bank has an associated SMMU_CBARn register. A Translation context bank of index *n* is associated with an SMMU_CBARn. This register provides additional attributes for the Translation context bank. This means a System MMU implementation must implement the same number of SMMU_CBARn registers as the number of implemented Translation context banks. Each SMMU_CBARn is part of the corresponding Translation context bank, but has a different access mechanism.

If SMMU_CBARn configures the associated Translation context bank for stage 1 translation, it can specify one of the following additional translation contexts:

- in an implementation that supports nested translation, a second Translation context bank
- Bypass mode, performing no stage 2 translation or address protection, but optionally downgrading the memory attributes from the stage 1 translation, in a similar way to the possible downgrading of memory attributes by a Stage 2 Translation context bank.

If a Translation context bank is configured to raise an interrupt in the event of a context fault, SMMU_CBARn specifies the interrupt to be asserted if the context fault occurs.

The SMMU_CBARn registers are organized as a table in the System MMU configuration address space. A configuration access to an unimplemented SMMU_CBARn register results in one of the following IMPLEMENTATION DEFINED behaviors:

- the access is treated as RAZ/WI
- the access generates a Configuration access fault, as Chapter 3 *The Fault Model* describes.

SMMU_CBARn.VMID specifies the Virtual Machine Identifier for the corresponding Translation context bank. When associated with a Non-secure, non-hypervisor Stage 1 Translation context bank, the VMID field is used as follows:

- The VMID is associated with a transaction being translated using this Translation context bank. The VMID is used for TLB tagging and matching purposes.
- The VMID is used for TLB maintenance operations issued in the corresponding Translation context bank. The VMID is used for TLB matching purposes.

See *SMMU_CBARn, Context Bank Attribute Registers on page 11-155* for more information.

Note

The use of VMID is modified for:

- Secure context banks. See *Interaction with the Security Extensions*
- Hypervisor-marked contexts. See *Effect of HYPC on TLBs on page 2-42*.

Interaction with the Security Extensions

In an implementation that includes the Security Extensions, reservation of a Translation context bank using SMMU_SCR1.NSNUMCBO also reserves the corresponding SMMU_CBARn register. This also determines the number of SMMU_CBARn registers visible to Non-secure software.

Secure software must only use *Stage 1 context with stage 2 bypass, type 0b01 on page 11-156*. Using any other type generates an invalid context fault.

A VMID can be associated with Secure transactions in both translation and bypass.contexts. This provides support in legacy systems where VMID is propagated to the endpoint in the system, as an attribute to the transaction, at which point further checks are made. When associated with a Secure transaction:

- a VMID has no involvement in translation matching
- it is IMPLEMENTATION DEFINED whether the VMID is visible in translation table walks and client transactions.

Reset state

On reset, the SMMU_CBARn. registers are not initialized. They must be initialized before use.

2.5.3 The translation tables

Translation tables specifying the output address and memory attributes for a context are stored in memory. The SMMU_CBn_TTBRm register, or registers, specify the translation table base address for the corresponding translation context, where *n* is 1 or 0. See:

- *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224*
- *SMMU_CBn_TTBR0, Translation Table Base Register on page 16-238*.

2.6 Translation and protection checks

If translation is indicated by the Context determination process, a translation table walk retrieves translation and memory attribute information for the transaction. The transaction might require two stages of translation, as configured in the Stream mapping table and SMMU_CBARn registers.

2.6.1 Translation table format

The translation table formats and translation process are mainly the same as those in the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition*.

Stage 2 translation table format

The processor architecture reserves bits [63:60] of the stage 2 translation table format for use by the System MMU.

Table 2-1 shows the System MMU architecture use of these bits.

Table 2-1 System MMU stage 2 translation table bits

PTE bits[63:60]	Name	Description	
63:62	WACFG	Write-Allocate Configuration	
		`0b00`	Use value supplied from previous stage
		`0b01`	Reserved
		`0b10`	Write-Allocate
		`0b11`	No Write-Allocate.
61:60	RACFG	Read-Allocate Configuration	
		`0b00`	Use value supplied from previous stage
		`0b01`	Reserved
		`0b10`	Read-Allocate
		`0b11`	No Read-Allocate.

The values `0b10` and `0b11` cause the allocation hint supplied as part of the transaction to be overridden with the specified value.

2.6.2 Domains

The System MMU translation process does not support the Domain feature of the ARM *Virtual Memory System Architecture* (VMSA) before the addition of the Large Physical Address Extension.

2.6.3 Implemented address size

The System MMU provides a number of IMPLEMENTATION DEFINED options for input and output address sizes, specified by the SMMU_IDR2.IAS field.

If an implementation has an input address size larger than 32 bits, a client transaction processed by a Stage 1 Translation context bank raises a translation fault if the input address bits above bit 31 do not have the value 0.

SMMU_IDR2.OAS specifies the implemented output address size.

2.6.4 Global translation table entries and the Security Extensions

In common with the processor architecture, the System MMU architecture supports global and non-global memory regions.

When using the Long-descriptor translation table format, the processor architecture can permit fetching of page tables from Non-secure memory when the processor is in Secure state. Similarly, the System MMU architecture permits Secure context banks to access Non-secure memory during lookups.

The System MMU architecture extends this behavior so that the first level descriptor can be fetched from Non-secure memory by using the NSCFG0 and NSCFG1 fields in SMMU_CBn_TTBCR. This behavior is also permitted when using the Short-descriptor format.

As with the processor architecture, a Secure translation must be treated as non-global, regardless of the value of the nG bit in the final descriptor, if either or both of the following apply:

- the NS bit of the descriptor is set to 1
- the descriptor is fetched from Non-secure memory, regardless of the mechanism that caused this behavior.

The purpose of this restriction is to provide a safety measure that prevents a Secure process from accidentally polluting the memory space of another process with a global entry that maps Non-secure memory.

2.6.5 High-level overview of the translation process

Figure 2-3 on page 2-36 shows an example of processing a transaction. Depending on IMPLEMENTATION DEFINED choices, other processing flows are possible. For example, StreamID indexing can replace StreamID matching.

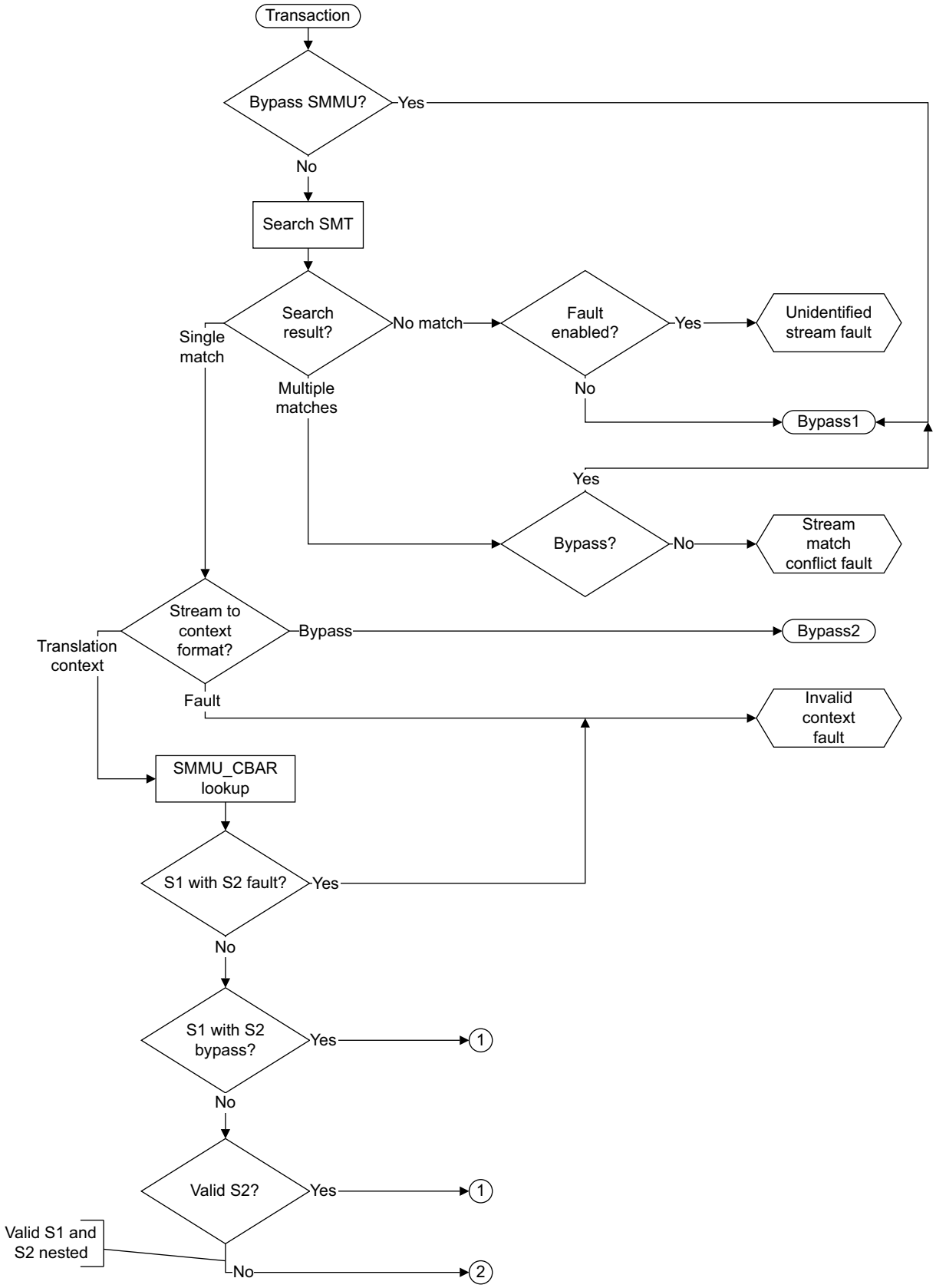


Figure 2-3 Example processing of a transaction from a Non-secure device

See *Interaction with the Security Extensions on page 2-30* for information about reserving Translation context banks.

SMMU_sCR0.CLIENTPD determines whether the transaction bypasses:

- System MMU translation
- protection checking
- attribute generation.

An instruction fetch by a Secure client is subject to a protection check based on the value of SMMU_SCR1.SIF. SMMU_sCR0.CLIENTPD controls whether other protection checks are bypassed.

If the transaction bypasses the System MMU:

- the address is not translated, which means that the output address is identical to the input address
- output attributes of the transaction are a function of the input attributes and SMMU_sCR0 fields.

If SMMU_sCR0.CLIENTPD indicates that the transaction is to be processed, the System MMU searches the Stream mapping table for a matching Stream mapping register group. See *The Stream mapping table on page 2-25*.

If no match is found, SMMU_sCR0.USFCFG determines how to handle the match failure. If bypass action is specified, the address is not translated, and the output attributes are the same as those for bypass at the earlier SMMU_sCR0.CLIENTPD stage.

If a search of the Stream mapping table yields multiple matches, SMMU_sCR0.SMCFCFG determines whether the transaction bypasses subsequent System MMU processing or incurs a Stream match conflict fault.

If the transaction is successfully matched in the Stream mapping table, an initial translation context format is determined using SMMU_S2CRn. If SMMU_S2CRn specifies Bypass mode:

- the address is not translated, which means that the output address is identical to the input address
- output attributes are a function of the input attributes and the SMMU_S2CRn fields.

If an initial translation context format is determined, a translation context is set where the input attributes to the translation process are the default input attributes and SMMU_S2CRn fields.

SMMU_CBARn defines additional configuration for the translation context. SMMU_CBARn.TYPE defines the register format.

For an outline of stage 1 with stage 2 bypass processing, or stage 2 processing, see Figure 2-4. Otherwise, see Figure 2-5 on page 2-38.

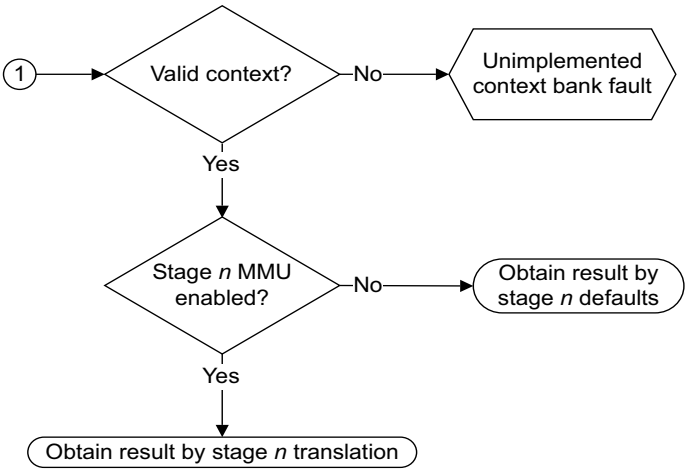


Figure 2-4 Stage 1 with stage 2 bypass, or stage 2

If the SMMU_CBARn lookup result is stage 1 with stage 2 bypass:

1. If the stage 1 context is valid, SMMU_CBn_SCTLR.M is read to see if the System MMU is enabled.

2. If the System MMU is enabled, a result is obtained from stage 1 translation tables and SMMU_CBARn. Otherwise, a result is obtained from default input attributes, stage 1 SMMU_CBn_SCTLR, SMMU_CBARn and SMMU_S2CRn.

If the SMMU_CBARn lookup result is stage 2:

1. SMMU_CBn_SCTLR.M controls whether the System MMU is enabled.
2. If the stage 2 context is valid and the System MMU is enabled, the result is obtained from the default input attributes and a stage 2 translation table lookup. Otherwise, the result is obtained from the default input attributes and stage 2 SMMU_CBn_SCTLR.

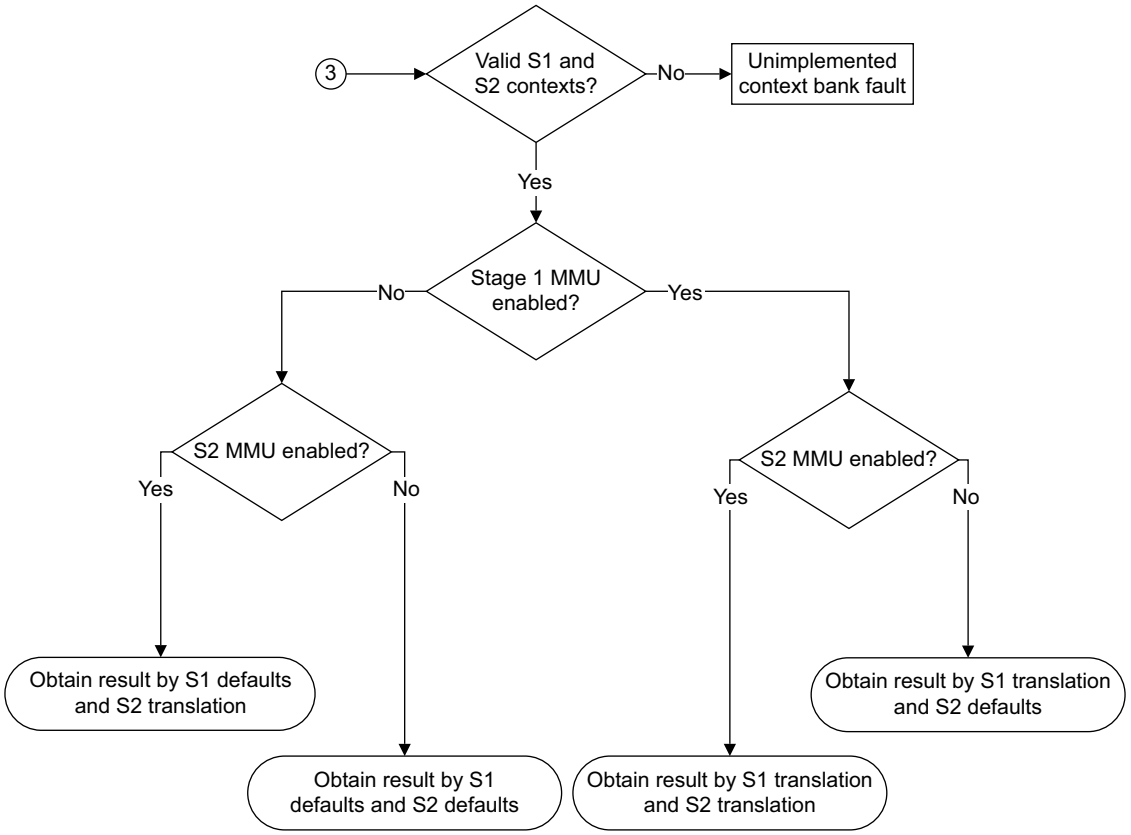


Figure 2-5 Stage 1 with stage 2 nested translation

If the SMMU_CBARn lookup result is stage 1 translation and stage 2 translation:

- The stage 1 SMMU_CBn_SCTLR.M bit controls whether the System MMU is enabled for stage 1 translation, and the stage 2 SMMU_CBn_SCTLR.M bit controls whether the System MMU is enabled for stage 2 translation.
- If the stage 1 and stage 2 contexts are valid, but the System MMU is not enabled for stage 1 translation:
  - if the System MMU is enabled for stage 2 translation, the result is obtained from the default input attributes, SMMU_S2CRn, the stage 1 SMMU_CBn_SCTLR fields, and a stage 2 translation table lookup
  - if the System MMU is not enabled for stage 2 translation, the result is obtained from the default input attributes, SMMU_S2CRn, the stage 1 SMMU_CBn_SCTLR fields, and the stage 2 SMMU_CBn_SCTLR fields.
- If the stage 1 and stage 2 contexts are valid, and the System MMU is enabled for stage 1 translation:
  - if the System MMU is enabled for stage 2 translation, the result is obtained from the stage 1 translation table lookup, and the stage 2 translation table lookup

— if the System MMU is not enabled for stage 2 translation, the result is obtained from the stage 1 translation table lookup and the stage 2 SMMU_CBn_SCTLR fields.

Memory type and shareability attributes

A transaction can pass through a number of steps as part of the System MMU translation process. The memory type and shareability attributes can be obtained by combining from a number of sources. If the final attribute for a memory location is Strongly-ordered or Device, the location is treated as Outer Shareable. See the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition* for information about shareability for Strongly-ordered and Device memory types.

In general, the final attributes used for a transaction depend on fields in several registers, in addition to the contents of the translation table. Table 2-2 shows, for each transaction type, whether each register field applies.

Table 2-2 Location of information for final memory and shareability attributes

Transaction	NSCFG	WACFG	RACFG	MemAttr	MTCFG	SHCFG	INSTCFG	PRIVCFG
Stage 1 translation table	Y	The *SMMU_CBn_MAIRm, Memory Attribute Indirection Registers on page 15-201* provide memory attribute selection.			-	Y	Partially determine permission fault	
Stage 2 translation table	-	Y[a]	Y[a]	Y	-	Y	Partially determine permission fault	
SMMU_S2CRn, bypass mode	Y	Y	Y	Y	Y	Y	-	-
SMMU_S2CRn, translation context	Y	Y	Y	Y	Y	Y	Y[b]	Y
Stage 1 SMMU_CBn_SCTLR[c]	Y	Y	Y	Y	Y	Y	-	-
Stage 2 SMMU_CBn_SCTLR[d]	-	Y	Y	Y	Y	Y	-	-
SMMU_CBARn[e]	-	Y	Y	Y	-	Y	-	-

a. Read Allocate and Write Allocate are not defined for processor stage 2 translation tables. However, bits are reserved for the SMMU for this purpose See Table 2-1 on page 2-34 for more information.
b. For reads only. It is IMPLEMENTATION DEFINED whether this applies to writes.
c. When Stage 1 context bank MMU behavior is disabled.
d. When Stage 2 context bank MMU behavior is disabled.
e. When used as Stage 2.

If any SMMU_sCR0 settings result in a transaction bypassing the Stream mapping table, as *Bypassing the Stream mapping table on page 2-26* describes, then SMMU_sCR0 specifies memory attribute transformation for the transaction.

Similarly, if the SMMU_S2CRn.TYPE field specifies that the initial translation context is Bypass mode, SMMU_S2CRn specifies memory attribute transformation for the transaction.

For translations that use a Stage 1 translation context bank, SMMU_S2CRn specifies the first memory attribute transformation, and then either:

- if the SMMU_CBn_SCTLR.M bit is set to 0, that is, context bank MMU behavior is disabled, SMMU_CBn_SCTLR specifies the next attribute transformation applied
- if the SMMU_CBn_SCTLR.M bit is set to 1, the translation table specifies the attributes.

When a Stage 1 descriptor specifies the attributes, this overrides all attributes except those used for permission checking, as defined by the IND and PNU bits in SMMU_CBn_FSYNRm. Therefore, the only SMMU_S2CRn fields that affect such transactions are INSTCFG and PRIVCFG.

The resultant attributes are then combined with the stage 2 attributes, depending on the context bank format that the SMMU_CBARn.TYPE field specifies:

- for Stage 1 context with stage 2 bypass format, the stage 1 context SMMU_CBARn specifies the stage 2 memory attributes
- for Stage 1 context with stage 2 context, the setting of the SMMU_CBn_SCTLR.M bit defines the mechanism for determining the stage 2 attributes. When the SMMU_CBn_SCTLR.M bit:
  - is set to 0, the stage 2 context SMMU_CBn_SCTLR specifies the stage 2 attributes
  - is set to 1, the translation tables specify the stage 2 attributes.

If SMMU_CBARn specifies the initial translation context to be a stage 2 translation context bank, the stage 1 attributes are treated as the incoming attributes after the SMMU_S2CRn transformation has been applied. The stage 2 attributes are considered to be created in the same way as those for the Stage 1 context with stage 2 context.

Stage 1 and stage 2 attributes are combined in accordance with ARMv7 architectural requirements. The stage 2 attributes can only strengthen memory types, where the memory types are listed strongest to weakest:

- Strongly-ordered
- Device
- Non-cacheable to normal memory
- Write-through to normal memory
- Write-back to normal memory.

Similarly, the following shareabilities are listed strongest to weakest:

- Outer Shareable
- Inner Shareable
- Non-shareable.

In the ARM architecture, stage 2 attributes do not include a read or write allocation hints. In the SMMU architecture when context bank translation is disabled, the stage 2 SMMU_CBn_SCTLR RACFG and WACFG fields can be used with stage 1 translation. If the stage 2 SMMU_CBn_SCTLR.M bit is set to 1 then there are extra fields in the descriptor that specify RACFG and WACFG fields that are used. See Table 2-1 on page 2-34 for more information.

For memory type and shareability attributes, the following behaviors are IMPLEMENTATION DEFINED:

Combining Read and Write hint allocations

It is IMPLEMENTATION DEFINED how read and write allocate hints from stage 2 attributes combine with stage 1 attributes.

Transforming bypassed transaction encodings

When the appropriate MTCFG field is set to 0, it is IMPLEMENTATION DEFINED whether certain bypassed client transactions have encodings that cannot be transformed. For example, an implementation might specify that incoming Strongly-ordered or Device transactions are not subject to any memory attribute transformation.

Note

In such cases the NSCFG field is guaranteed to be applied if security state determination classes the transaction as Secure. If the appropriate MTCFG field is set to 1, the transformation is always applied, regardless of the encoding of the incoming transaction.

Instruction fetch configuration

It is IMPLEMENTATION DEFINED whether a SMMU_S2CRn.INSTCFG field transformation is applied to writes if that transformation would convert the transaction to an instruction write.

2.7 Hypervisor-marked contexts

If the implementation supports stage 1 translation, that is, if the SMMU_IDR0.S1TS bit is set to 1, hypervisor context (HYPC) is available.

Translation banks for the hypervisor context are called HYPC banks, or hypervisor banks, and are intended to be used by the hypervisor for translating devices that it owns and are operating on its own behalf. Hypervisor context applies when the SMMU_CBARn.HYPC bit is set to 1.

Note

The register formats do not permit HYPC banks to be nested.

2.7.1 Architectural effects of HYPC

The effects of HYPC in the SMMU is analogous to the ARMv7 PL2 control of Non-secure access privileges that *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition* describes. Table 2-3 shows the applicable address translation registers.

Table 2-3 SMMU and ARMv7 hypervisor registers used for address translation

SMMU register	ARMv7 register	Notes
SMMU_CBn_TTBR0	HTTBR	SMMU_CBn_TTBR1 is ignored. The SMMU_CBn_TTBR0.ASID field is ignored The following fields have no effect: • SMMU_CBn_SCTLR.AFE • SMMU_CBn_SCTLR.TEX. • SMMU_CBARn.VMID In the Long-descriptor Table descriptor, the APTable[0], PXNTable, and PXN bits are reserved, SBZ.
SMMU_CBn_TTBCR	HTCR	The SMMU_CBn_TTBCR.EAE bit is ignored and considered to be set to 1, indicating stage 1 Long-descriptor format.
SMMU_CBn_MAIRm	HMAIRn	Includes register 0 and register 1 in each case.

Because there is no separation of privileged and unprivileged execution in HYPC, the Stage 1 address translation commands in Table 4-1 on page 4-65 might result in different behavior. In any SMMU implementation that uses HYPC:

- the following commands are guaranteed to work correctly:
  - SMMU_CBn_ATS1PR
  - SMMU_CBn_ATS1PW
- the following commands might work correctly:
  - SMMU_CBn_ATS1UR
  - SMMU_CBn_ATS1UW.

When targeting a context bank marked as HYPC, the global Stage 1 address translation operations in Table 4-2 on page 4-68 might result in different behavior, and:

- the following commands are guaranteed to work correctly:
  - SMMU_sGATS1PR
  - SMMU_sGATS1PW
- the following commands might work correctly:
  - SMMU_sGATS1UR
  - SMMU_sGATS1UW.

2.7.2 Effect of HYPC on TLBs

TLB entries that are allocated as a result of HYPC accesses are Hyp tagged. All Hyp tagged TLB entries are considered to belong to the same software entity and have no associated ASID or VMID.

2.7.3 Interaction with the Security Extensions

In the ARM architecture, a hypervisor exists only in Non-secure state. In the SMMU architecture, there is no requirement for HYPC banks to store and check security state, that is, there is no architectural definition of Secure or Non-secure Hyp tagged TLB entries.

It is therefore possible for a Secure HYPC bank to encounter a TLB entry allocated by a Non-secure HYPC bank, resulting in a security violation. To prevent this, Secure software must either:

- avoid using HYPC contexts, by ensuring that the SMMU_CBARn.HYPC bit is set to 0 for all Secure context translation banks
- prevent Non-secure software from configuring a bank as HYPC, by ensuring that:
  - the SMMU_SCR1.GASRAE bit is set to 1, to prevent Non-secure access to SMMU_CBARn
  - a Non-secure bank is never configured with SMMU_CBARn.HYPC bit set to 1.

2.7.4 Effect of permission checking and bus marking

Depending on the bus standard, a hypervisor privilege tag might be included in the transaction.

If a privilege tag is not included, then the permission model for HYPC banks is independent of the bus tag. For example, a bus might only transport the concept of privileged and unprivileged, and the permission model of a HYPC bank is independent of these attributes.

Chapter 3
The Fault Model

This chapter describes System MMU fault handling. It contains the following sections:

3.1 Overview of fault types

A System MMU implementation might encounter any of the following types of fault:

- a fault on a Translation context bank:
  - translation fault
  - permission fault
  - external fault.
- a global fault:
  - Invalid context fault
  - Unidentified stream fault
  - Stream match conflict fault
  - Unimplemented context bank fault
  - Unimplemented context interrupt fault
  - Configuration access fault
  - permission fault
  - external fault.

The System MMU provides resources to record, process and report these faults.

For details about:

- faults on a Translation context bank, see *Context faults on page 3-51*
- global faults, see *Global faults on page 3-56*.

3.2 Fault-handling terminology

The following terminology applies to faults:

encounter A System MMU *encounters* a fault as a result of either:

- processing a transaction in the System MMU
- a transaction response returned to the System MMU by an external agent.

record A System MMU *records* a fault by:

- The appropriate fault status register capturing fault status information. See:
  - *SMMU_sGFSR, Global Fault Status Register on page 10-131*
  - *SMMU_CBn_FSR, Fault Status Register on page 15-197*.
- The corresponding fault address register capturing the fault address. See:
  - *SMMU_sGFAR, Global Fault Address Register on page 10-130*
  - *SMMU_CBn_FAR, Fault Address Register on page 15-197*.
- One or more fault syndrome registers capturing additional details. See:
  - *SMMU_sGFSYNR0, Global Fault Syndrome Register 0 on page 10-133*
  - *SMMU_sGFSYNR1, Global Fault Syndrome Register 1 on page 10-134*
  - *SMMU_sGFSYNR2, Global Fault Syndrome Register 2 on page 10-135*
  - *SMMU_CBFRSYNRAn, Context Bank Fault Restricted Syndrome Register A on page 11-161*
  - *SMMU_CBn_FSYNRm, Fault Syndrome Registers on page 15-199*.

report A System MMU *reports* a fault by returning an error to the initiator of the transaction that caused the fault. Alternatively, an external system component processing a transaction issued by the System MMU might *report* a fault to the System MMU.

Support for fault reporting in the initiating device and in the interconnect between the device and the System MMU is IMPLEMENTATION DEFINED.

For exclusive access transactions, In the context of the SMMU, reporting means specifically reporting an exclusive access transaction as aborting.

interrupt A System MMU might assert an interrupt signal because of a fault.

client A *client* access is a memory transaction issued by a client device, that the System MMU is to process.

configuration A *configuration* access is a memory transaction issued by a device that accesses a register or a command in the System MMU configuration address space.

3.3 Handling multiple memory faults

The ARM processor architecture only has to handle a single memory fault at any given time. In contrast, a System MMU implementation is a shared resource that might receive simultaneous memory access requests from multiple agents. Such concurrency means that the System MMU could encounter multiple faults in a short period of time, or could encounter a second fault before the first fault is acknowledged by software.

For a fault on a Translation context bank, SMMU_CBn_FSR.MULTI==1 if a fault is encountered when this register already has information about a previous fault that has not yet been acknowledged by software. Similarly, SMMU_sGFSR.MULTI in the case of a global fault. In both cases, the register records full details of the first fault, and only the Multiple Fault Status flag is set on a subsequent fault.

For details specific to translation context faults, see *Multiple faults on page 3-52*. For details specific to global faults, see *Multiple fault conditions on page 3-57*.

3.4 Recording memory attributes

Fault syndrome registers record the memory attributes of a transaction that caused a fault, as follows:

- for a fault on a Translation context bank, SMMU_CBn_FSYNRm
- for a global fault, SMMU_sGFSYNR0.

The System MMU architecture defines specific handling of the following memory attributes:

- *Instruction, not Data* (IND)
- *Privileged, not Unprivileged* (PNU)
- *Non-secure Attribute* (NSATTR).

The handing of these attributes is:

- If a global fault is recorded, the default input attributes IND, PNU and NSATTR are recorded.
- If a stage 1 or a stage 2 context fault is recorded, the memory attributes IND, PNU and NSATTR, as presented to the System MMU *after* Context determination, are recorded. This means that the values of IND, PNU and NSATTR are possibly overridden by one of:
  - SMMU_S2CRn.INSTCFG
  - SMMU_S2CRn.PRIVCFG
  - SMMU_S2CRn.NSCFG.

ARM recommends that other memory attributes recorded in the fault syndrome registers are handled in the same way as the IND, PNU and NSATTR attributes, although the handling of other memory attributes is IMPLEMENTATION DEFINED.

3.5 Recording nested translation faults

For a nested translation, that is translations that use a stage 1 context followed by a stage 2 context, faults encountered at stage 2 are recorded either in the stage 2 context bank or in the *SMMU_sGFSR, Global Fault Status Register on page 10-131*.

The following fault types are recorded in the stage 2 context bank:

- stage 2 permission faults
- stage 2 translation faults
- external aborts.

The following fault types are recorded in the *SMMU_sGFSR, Global Fault Status Register on page 10-131*:

- stage 1 or stage 2 invalid context fault
- Unimplemented context bank fault
- Unimplemented context interrupt fault.

Note

Software executing in global register space must not transfer control of a context bank to software executing at a lower privilege until the bank is fully configured. For example, this might apply to a hypervisor transferring control to a Guest OS. This can result in low privilege software affecting accesses to SMMU_sGFSR by higher privilege software.

3.5.1 Distinction of Stage 1 nested faults and Stage 2 only faults

There is no simple mechanism for a hypervisor to establish whether a fault in a stage 2 context bank is either:

- a result of a stage 1 nested client transaction faulting in stage 2, in which case:
  - SMMU_CBn_FAR contains the VA of the client transaction
  - the SMMU_CBn_FSYNR0.S1PTWF bit indicates a fault on the stage 1 translation table walk
- a result of a client transaction mapping directly to the stage 2 bank, in which case SMMU_CBn_FAR contains the IPA of the client transaction

Software must therefore be capable of distinguishing such differences, either by:

- using separate stage 2 banks for the nested and non-nested cases
- matching SMMU_CBFRSYNRAn with the Stream Match Register groups to determine the context.

3.5.2 Behavior of stage 2 faults within nested translations

This section describes behaviors associated with fault handling for nested client and address translation operations, where a fault occurs during stage 2 translation. The behavior depends on whether stage 2 transactions are configured to terminate or to stall on a fault, as *Handling a Context fault on page 3-53* describes.

Behavior when stage 2 transactions terminate on a fault

If the client transaction is terminated when a stage 2 fault occurs, no fault is recorded in stage 1, even for a stage 1 translation table walk that faults. However, depending on the setting of stage 2 SMMU_CBn_SCTLR.CFRE bit, the transaction might report an abort to the client device.

If a stage 2 fault is encountered during an address translation operation initiated in a Stage 1 Translation context bank, the fault is always recorded as *Address Translation Operation Terminated* (ATOT) in the stage 1 SMMU_CBn_PAR. See *Fault handling within nested translation operations initiated in a context bank on page 4-65*.

If an outstanding stage 2 context fault exists and the stage 2 SMMU_CBn_SCTLR.HUPCF bit is set to 0, then all subsequent transactions to that context bank result in a fault, regardless of whether the operation might have completed successfully

If a transaction aborts in stage 2 and the stage 2 SMMU_CBn_SCTLR.HUPCF bit is set to 1, then all subsequent transactions abort. This applies to both client transactions and address translation operations.

Note

This is very different from the behavior for non-nested translations, where each transaction in the downstream memory system aborts independently.

Behavior when stage 2 transactions stall on a fault

If stage 2 transactions are configured to stall when a fault occurs, the hypervisor has more control than it does for transactions configured to terminate. The hypervisor is not required to ensure that all IPAs for a device have valid translations, and can:

- make translations available on a Stage 2 fault that stalls
- use stage 2 access flags to discover faults
- inject synchronous faults manually into stage 1 translation.

3.6 Fault interrupts

A System MMU can communicate faults to software using the following types of interrupt:

- Global interrupt
- Context interrupt.

3.6.1 Global interrupts

A System MMU implements at least the following logical Global interrupts:

- SMMU_NsgCfglrpt
- SMMU_NSglrpt.

SMMU_NsgCfglrpt is used when a Configuration Access Fault is recorded. SMMU_NSglrpt is used when any other type of global fault is recorded.

In an implementation that includes the Security Extensions, the System MMU also implements SMMU_gCfglrpt and SMMU_glrpt. These are the Secure equivalents of SMMU_NsgCfglrpt and SMMU_NSglrpt respectively.

3.6.2 Context interrupts

A Context interrupt is raised for a fault on a Translation context bank. SMMU_IDR0.NUMIRPT specifies the IMPLEMENTATION DEFINED number of Context interrupts supported by the System MMU.

SMMU_CBARn.IRPTNDX specifies the interrupt number to assert in the event of a Translation context bank fault.

ARM recommends that:

- SMMU_CBARn.IRPTNDX is software configured in the range specified by SMMU_IDR0.NUMIRPT
- if Secure software gives Non-secure software at least one Translation context bank, the Secure software provides the Non-secure software with at least one Context interrupt.

An Unimplemented context interrupt fault occurs when all of the following conditions hold true:

- SMMU_CBARn.IRPTNDX is configured outside the range reported by SMMU_IDR0.NUMIRPT
- stage 1 SMMU_CBn_SCTLR.CFIE or stage 2 SMMU_CBn_SCTLR.CFIE is 1
- a context fault causes the assertion of a Context interrupt.

In an implementation that includes the Security Extensions, SMMU_SCR1.NSNUMIRPTO reserves Context interrupts for Secure software.

An SMMU_CBARn register reserved by Secure software must only specify a Context interrupt reserved by Secure software. Otherwise, UNPREDICTABLE behavior results.

An SMMU_CBARn register not reserved by Secure software can only specify a Context interrupt not reserved by Secure software. Otherwise, an Unimplemented context interrupt fault could occur.

3.6.3 Interrupt assertion

A System MMU interrupt is asserted in response to a fault recorded in the Fault Status Register, providing interrupts are enabled. See SMMU_CBn_FSR and SMMU_sGFSR.for details.

An interrupt is asserted regardless of whether the Fault Status Register is set by a fault or by a non-zero value written to SMMU_sGFSRRESTORE or SMMU_CBn_FSRRESTORE.

A value of 1 written to any non-reserved bit in SMMU_CBn_FSR or SMMU_sGFSR clears that bit. A value of 0 written to any of these bits leaves the bit unchanged.

When all of the relevant bits in a Fault Status Register are cleared, the corresponding interrupt is deasserted, providing no other Fault Status Register is contributing to the assertion of the interrupt.

3.7 Context faults

A context fault is associated with a particular Translation context bank. The following types of context fault exist:

- translation fault
- permission fault
- external fault.

A translation fault occurs if the System MMU does not obtain a translation for a transaction.

A permission fault occurs if the System MMU retrieves a translation for a transaction, but the transaction has insufficient privileges to reach completion.

An external fault occurs if an external abort is reported to the System MMU during transaction processing.

3.7.1 *Secure Instruction Fetch* (SIF) permission faults

In an implementation that includes the Security Extensions, Secure instruction fetches can result in permission faults when the *Secure Instruction Fetch* (SIF) bit in SMMU_SCR1 is set to 1. A permission fault results when a Secure domain client access attempts to exit as a Non-secure instruction *after* any transformation indicated by the SMMU_S2CRn.INSTCFG bit has been applied.

For transactions associated with a particular Translation context bank, a SIF permission fault is recorded using the SMMU_CBn_FSR.PF bit when all of the following apply:

- the SMMU_SCR1.SIF bit is set to 1
- the transaction has been routed to a Secure context bank
- the transaction is classified as Instruction, *after* applying any SMMU_S2CRn.INSTCFG transformation
- the outgoing transaction is Non-secure.

This check is also applied to transactions that bypass the Stream mapping table. See *Bypassing the Stream mapping table on page 2-26* for more information.

Note

For client transactions that use stage 1 or stage 2 context resources, the instruction fetch configuration attribute specified by the SMMU_S2CRn. INSTCFG bit can override the IND, PNU and NSATTR memory attributes. See *Stream-to-Context Register, SMMU_S2CRn on page 2-27*. When referring to instruction fetches in this document, it is assumed that any such overrides are accounted for.

3.7.2 Recording a context fault

The registers that record information about a transaction causing a fault on a Translation context bank are:

- SMMU_CBn_FSR
- SMMU_CBn_FAR
- SMMU_CBn_FSYNRm.

SMMU_CBFRSYNRAn records additional fault syndrome details about the fault, where *n* is the index into the Translation context bank.

Two-stage translation faults

A context fault is only recorded for one stage of translation. This means that in an implementation supporting two translation stages:

- if no stage 2 fault is encountered, a context fault that occurs is recorded for the stage 1 translation context
- if a stage 2 fault is encountered, the fault is recorded for the stage 2 translation context.

External faults

If an external fault is reported to the System MMU in response to a fetch issued as part of a translation table walk, the fault is recorded synchronously in the Translation context bank. If any other external fault is reported to the System MMU in response to a transaction associated with a Translation context bank, it is IMPLEMENTATION DEFINED whether the fault is:

- recorded synchronously
- recorded asynchronously
- not recorded.

Synchronous recording of an external fault updates the following registers:

- SMMU_CBn_FSR
- SMMU_CBn_FAR
- SMMU_CBn_FSYNRm
- SMMU_CBFRSYNRAn.

Asynchronous recording of an external fault updates the following registers:

- SMMU_CBn_FSR
- SMMU_CB*n*_FSYNR0.AFR is set to 1, as *SMMU_CBn_FSYNRm, Fault Syndrome Registers on page 15-199* describes.
- for an external fault on a translation table walk, SMMU_CB*n*_FSYNR0.PTWF is set to 1. Otherwise, it is cleared to 0, as *SMMU_CBn_FSYNRm, Fault Syndrome Registers on page 15-199* describes.
- other fields in the following registers are UNK/SBZP:
  - SMMU_CBn_FSYNRm
  - SMMU_CBn_FAR.

For more information about external faults, see *External faults on page 3-60*.

Multiple faults

SMMU_CBn_FSR.MULTI indicates the presence of multiple outstanding faults.

If a fault is encountered when all of the fields in SMMU_CBn_FSR are zero, including SMMU_CBn_FSR.MULTI, the following registers record full details of the fault:

- SMMU_CBn_FSR
- SMMU_CBn_FAR
- SMMU_CBn_FSYNRm.

If a fault is encountered when SMMU_CBn_FSR is non-zero:

- the SMMU_CBn_FSR.MULTI bit is set to 1
- details of the fault are *not* recorded.

Context Bank Fault Restricted Syndrome Register, SMMU_CBFRSYNRA*n*

There is an SMMU_CBFRSYNRAn register for each Translation context bank. *n*, the Translation context bank index, denotes the matching SMMU_CBFRSYNRAn register in the range 0-127. The register provides information about the fault recorded in the SMMU_CBn_FSR register of a stage 1 or a stage 2 Translation context bank.

The information recorded in SMMU_CBFRSYNRAn could present a virtualization hole if the register were to reside in the Translation context bank address space, potentially accessible by a Guest OS. Therefore, each SMMU_CBFRSYNRAn register is in the global address space. For information about the Virtualization Extensions, including what is meant by a Guest OS in this context, see the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition*.

The SMMU_CBFRSYNRAn registers are organized as a table. Because a SMMU_CBFRSYNRAn register exists for each Translation context bank, a System MMU implementation must implement the same number of SMMU_CBFRSYNRAn registers as the number of Translation context banks. This means that the number of SMMU_CBFRSYNRAn registers implemented must match the number of Translation context bank table entries.

A configuration access to an unimplemented SMMU_CBFRSYNRAn register results in either of the following IMPLEMENTATION DEFINED behaviors:

- the access is RAZ/WI
- a Configuration access fault.

For more information about Configuration access faults, see *Configuration access on page 3-59*.

In an implementation that includes the Security Extensions, SMMU_SCR1.NSNUMCBO reserves a Translation context bank and an SMMU_CBFRSYNRAn register for that Translation context bank. The number of SMMU_CBFRSYNRAn registers visible to Non-secure software adjusts accordingly.

SMMU_IDR1.NUMCB indicates the number of implemented SMMU_CBFRSYNRAn registers.

3.7.3 Handling a Context fault

A System MMU handles a Context fault by either terminating or stalling the transaction that caused the fault.

If the System MMU terminates the fault, there is no memory access. The access behaves as RAZ/WI. Depending on the value of the stage 1 or stage 2 SMMU_CBn_SCTLR.CFRE field, the System MMU reports the fault to the initiator of the faulty transaction. See *Context fault interrupts on page 3-55* for more information.

If the System MMU stalls the fault, software can subsequently write to SMMU_CBn_RESUME to retry or terminate the stalled transaction. The SMMU_CBn_FSR.SS field is cleared as a result of invoking the operation to resume the transaction.

The stage 1 SMMU_CBn_SCTLR.CFCFG and stage 2 SMMU_CBn_SCTLR.CFCFG registers handle context faults on a per context basis.

It is IMPLEMENTATION DEFINED whether the System MMU supports the stall mode of operation.

The Stall fault model

If a fault is encountered by the System MMU when the Stall fault model is enabled, the System MMU stalls the transaction that caused the fault, pending action to rectify the cause.

ARM recommends that the Stall fault model generally be used with fault interrupting. This is so that a supervisory agent can be signaled to rectify the cause and restart the transaction.

Stage 1 SMMU_CBn_SCTLR.CFCFG and stage 2 SMMU_CBn_SCTLR.CFCFG enable the Stall fault model, per translation context.

It is IMPLEMENTATION DEFINED whether a System MMU implementation includes the Stall fault model. If an implementation does not include the Stall fault model, stage 1 SMMU_CBn_SCTLR.CFCFG and stage 2 SMMU_CBn_SCTLR.CFCFG each have a fixed value and are RAZ/WI.

Stage 1 SMMU_CBn_SCTLR.HUPCF and stage 2 SMMU_CBn_SCTLR.HUPCF provide more configuration options for the Stall fault model, as follows:

- If stage 1 SMMU_CBn_SCTLR.HUPCF or stage 2 SMMU_CBn_SCTLR.HUPCF is 0 and a fault occurs, no more transactions for that Translation context bank are processed until the fault is cleared.
- If stage 1 SMMU_CBn_SCTLR.HUPCF or stage 2 SMMU_CBn_SCTLR are 1 and a fault occurs, more transactions can be processed for that Translation context bank before the original fault is cleared. See *Hit-under-fault on page 3-54* for more information.

  It is IMPLEMENTATION DEFINED whether a System MMU implementation includes functionality for processing a transaction under an existing fault. If the implementation does not provide this functionality:

  — the behavior is identical to stage 1 SMMU_CBn_SCTLR.HUPCF or stage 2 SMMU_CBn_SCTLR.HUPCF being 0

— stage 1 SMMU_CBn_SCTLR.HUPCF and stage 2 SMMU_CBn_SCTLR.HUPCF remain writable.

The number of transactions processed after the original faulty transaction is IMPLEMENTATION DEFINED.

The number of subsequent transactions that can raise a fault before no more transactions are processed for that Translation context bank until the original fault condition is cleared, is also IMPLEMENTATION DEFINED.

Depending on the System MMU implementation, system topology, and devices connected to the System MMU, it might not be possible to guarantee that a transaction in one translation context is unaffected by a stalled transaction in another context. Therefore, use of the Stall fault model in a translation context might not be appropriate.

SMMU_sCR0.STALLD can be configured to globally disable the Stall fault model. This forces each stage 1 SMMU_CBn_SCTLR.CFCFG or stage 2 SMMU_CBn_SCTLR.CFCFG to be RAZ/WI.

In an implementation that includes the Security Extensions:

- SMMU_CR0.STALLD must apply to Non-secure contexts banks, and can optionally apply to Secure Translation context banks
- SMMU_SCR0.STALLD provides an override bit that has equivalent behavior to SMMU_CR0.STALLD.

SMMU_sCR0.STALLD is expected to be configured as part of the reset initialization process.

In response to a stalled context fault, if the System MMU is configured to raise an interrupt, supervisory software typically performs one of the following actions:

- Fixes the faulty translation and retries processing the stalled transaction by writing the appropriate value to SMMU_CBn_RESUME. Fixing the faulty transaction might involve updating translation tables and TLB maintenance.
- Terminates the stalled transaction by writing the appropriate value to SMMU_CBn_RESUME. A transaction terminated in this way returns no data for reads, and writes are ignored.

  Depending on the setting of stage 1 SMMU_CBn_SCTLR.CFRE or stage 2 SMMU_CBn_SCTLR.CFRE, the System MMU can report an abort to the initiator of the transaction. See *Context fault interrupts on page 3-55*.

  Fault recording does not occur for a transaction terminated by the SMMU_CBn_RESUME operation. This is because the fault is logged when the transaction first stalls.

Hit-under-fault

A System MMU can optionally continue to process a transaction before the condition causing an existing fault is rectified. For a translation context, Hit-under-fault behavior, if enabled, means that any subsequent transaction mapped to that context is processed, regardless of whether an outstanding fault is recorded for that context.

The following types of behavior apply:

- If termination behavior is selected for a translation context and a fault is active for that context, each subsequent transaction is processed separately. If a fault is raised on a subsequent transaction, it terminates and SMMU_CBn_FSR records a multiple fault condition.
- If stall behavior is selected for a translation context and a fault is active for that context, each subsequent transaction is processed separately. If a fault is raised on a subsequent transaction, that transaction waits, no fault status information is recorded on it, and it is retried after the original fault condition clears.

If Hit-under-fault is not enabled, any subsequent transaction mapped to that translation context is processed the same as the original faulty transaction, as follows:

- If termination behavior is selected for a translation context and a fault is active for that context, each subsequent transaction terminates. SMMU_CBn_FSR does *not* record any fault condition for any subsequent transaction beyond the original transaction. The termination of a subsequent transaction is regarded as a side-effect of the original fault.
- If stall behavior is selected for a translation context and a fault is active for that context, any subsequent transaction stalls, and only resumes after the original fault condition is cleared.

Stage 1 SMMU_CBn_SCTLR.HUPCF and stage 2 SMMU_CBn_SCTLR.HUPCF adjust the Hit-under-fault model.

3.7.4 Context fault interrupts

A System MMU can be configured to respond to a context fault by raising an interrupt. Stage 1 SMMU_CBn_SCTLR.CFIE and stage 2 SMMU_CBn_SCTLR.CFIE determine whether to raise an interrupt using the context interrupt output associated with the Translation context bank. The SMMU_CBARn.IRPTNDX field associated with the Translation context bank specifies which context interrupt to use. The value configured in this field is expected to be consistent with the number of context interrupts reported by SMMU_IDR0.NUMIRPT.

An Unimplemented context interrupt fault occurs if:

- IRPTNDX is configured outside the range reported by SMMU_IDR0.NUMIRPT
- stage 1 SMMU_CBn_SCTLR.CFIE or stage 2 SMMU_CBn_SCTLR.CFIE have the value 1
- aside from the value in IRPTNDX, a context fault in the associated Translation context bank would result in the assertion of a context interrupt.

3.7.5 Reporting a Context fault

The System MMU can be configured to report a Context fault. The value of stage 1 SMMU_CBn_SCTLR.CFRE or stage 2 SMMU_CBn_SCTLR.CFRE indicates whether an abort is reported to the initiator of a terminated transaction.

3.7.6 Enabling Global stall mode

The System MMU might give the ability to globally stall the processing of each translation that arrives after a stall fault is raised in a Translation context bank. The global stall applies to all transactions arriving at the System MMU, regardless of which translation context they map to.

SMMU_sCR0.GSE enables global stalling.

It is IMPLEMENTATION DEFINED whether global stall is supported:

- In an implementation that does not support global stalling, SMMU_sCR0.GSE is RAZ/WI.
- In an implementation that supports global stalling, SMMU_sCR0.STALLD can disable global stalling. When SMMU_sCR0.STALLD is 1, SMMU_sCR0.GSE is RAZ/WI.

In an implementation that includes the Security Extensions:

- SMMU_CR0.GSE must apply to Non-secure Translation context banks, and an implementation is permitted to apply the setting of SMMU_CR0.GSE to transactions processed by Secure Translation context banks.
- SMMU_SCR0.GSE provides functionality that is equivalent to the functionality provided by SMMU_CR0.GSE.
- SMMU_CR0.STALLD must apply to SMMU_CR0.GSE, and can optionally apply to SMMU_SCR0.GSE. Equivalent requirements exist for SMMU_SCR0.STALLD.

3.8 Global faults

A global fault is a fault that occurs when either:

- a transaction being processed by the System MMU has no associated Translation context bank
- a Translation context bank is not the appropriate place to record the fault.

A global fault can occur because of:

- An Invalid context fault, where the fault context has been selected in either:
  - the Stream mapping register group that matches the transaction
  - the SMMU_CBARn register of the initial context specified by the Stream Mapping process.
- An Unidentified stream fault, where both:
  - no match for the transaction is found in the stream matching registers
  - SMMU_sCR0.USFCFG has enabled the relevant fault reporting.
- A Stream match conflict fault, where multiple matches for a transaction are detected in the stream matching registers and SMMU_sCR0.SMCFCFG has enabled the relevant fault reporting capability.
- An Unimplemented context bank fault, where the transaction maps to a an unimplemented translation context. For example, in an implementation that does not include stage 2 nested translation, an attempt to use a Stage 2 Translation context bank might result in an Unimplemented context bank fault.
- An Unimplemented context interrupt fault, where a transaction causes a fault in a Translation context bank, and that translation context has been configured to assert a context interrupt that is not implemented.
- A Configuration access fault, where a configuration access is made to a non-existent System MMU configuration register.
- A Permission fault, where a transaction has insufficient permission to be processed.
- An external fault, where an external abort has been reported during the processing of a transaction.

3.8.1 Recording a global fault

The following registers record information about a transaction that causes a global fault:

- SMMU_sGFSR
- SMMU_sGFAR
- SMMU_sGFSYNR0
- SMMU_sGFSYNR1.

For Configuration access faults, the following conditions apply:

- SMMU_sGFSR is updated
- SMMU_sGFSYNR0 is updated, with the following exceptions:
  - NESTED is captured as the value 0
  - NSSTATE is UNK.
- SMMU_sGFAR is updated
- SMMU_sGFSYNR1 is UNK.

For external faults, the following conditions apply:

- SMMU_sGFSR is updated
- SMMU_sGFAR, SMMU_sGFSYNR0 and SMMU_sGFSYNR1 are UNK.

Interaction with the Security Extensions

In an implementation that includes the Security Extensions, SMMU_sGFSR is Banked for security. The security state of the transaction causing the global fault determines which banked SMMU_sGFSR register has information about the transaction.

Multiple fault conditions

The Global Fault Status Registers record details of one outstanding fault. If a subsequent fault occurs before an outstanding fault is serviced, the subsequent fault is recorded, but without the full details that accompany the initial fault. SMMU_sGFSR.MULTI indicates the existence of multiple outstanding faults recorded in SMMU_sGFSR.

If a global fault is encountered when SMMU_sGFSR is 0, including SMMU_sGFSR.MULTI, the fault status and syndrome details are recorded in:

- SMMU_sGFSR
- SMMU_sGFAR
- SMMU_sGFSYNR0
- SMMU_sGFSYNR1.

If a global fault occurs when SMMU_sGFSR is non-zero, including SMMU_sGFSR.MULTI, the SMMU_sGFSR.MULTI bit is set to 1 and no other fault status or syndrome state is updated.

Two-stage faults

SMMU_sGFSYNR0.NESTED provides information for the following types of fault in SMMU_sGFSR:

- Invalid context fault
- Unimplemented context bank fault
- Unimplemented context interrupt fault.

The NESTED bit indicates whether the fault condition occurred as a result of either:

- the initial translation context specified by the stream mapping process in SMMU_S2CRn
- a stage 2 nested translation context specified in the SMMU_CBARn register of the initial context.

Permission faults

Secure instruction fetch transactions can be subject to a protection check. If the SMMU_SCR1.SIF bit is set to 1, a permission fault is recorded when a Secure domain access attempts to exit the SMMU as a Non-secure instruction.

This check applies globally, and if not associated with a Translation context bank records faults to SMMU_SGFSR. Otherwise, faults are recorded in the associated SMMU_CBn_FSR.

3.8.2 Handling a Global fault

A transaction is terminated by the System MMU on encountering a global fault. The accesses behave as RAZ/WI. The appropriate fault status register is updated.

3.8.3 Reporting a Global fault

A System MMU can be configured to report a global fault by:

- reporting client transactions if SMMU_sCR0.GFRE is enabled
- reporting configuration transactions if SMMU_sCR0.GCFGFRE is enabled.

Global fault interrupts

A System MMU can be configured to use the following interrupt mechanisms in response to encountering a global fault:

- SMMU_NSgCfgIrpt, to indicate that a Configuration Access Fault has been recorded, where SMMU_sCR0.GCFGFIE enables reporting of the fault. SMMU_sGFSR.CAF specifies whether a Configuration Access Fault has been recorded.
- SMMU_NSgIrpt, to indicate whether any other global fault has been recorded, where SMMU_sCR0.GFIE enables reporting of the fault. SMMU_sGFSR specifies whether such a fault has been recorded.

Note

If SMMU_sGFSR.MULTI is 1 because of a Configuration access fault, SMMU_NSglrpt is asserted instead of SMMU_NSgCfglrpt.

In an implementation that includes the Security Extensions, the interrupt for communicating these fault classes depends on which Banked copy of SMMU_sGFSR is updated. If the global fault updates SMMU_sGFSR, interrupt signaling is as previously stated. If the global fault updates SMMU_sGFSR:

- SMMU_gCfgIrpt indicates Configuration access faults. SMMU_SCR0.GCFGFIE enables its assertion.
- SMMU_gIrpt indicates all other global faults. SMMU_SCR0.GFIE enables its assertion.

See *SMMU_sCR0, Configuration Register 0 on page 10-120* for more information.

Global faults and untranslated address operations

Client transaction classes that are not subject to translation might be processed by the System MMU. An untranslated address operation might be a barrier transaction, or a TLB maintenance operation that does not specify an address.

When a System MMU processes such a transaction, the transaction remains subject to the stream mapping process. Therefore, any of the following types of fault can occur:

- Invalid context fault
- Unidentified stream fault
- Stream match conflict fault
- Unimplemented context bank fault.

Inclusion of fault reporting in the initiating device and the interconnect between the device and the System MMU is IMPLEMENTATION DEFINED.

3.9 Configuration access

System MMU configuration is performed through a register space in the system address map. This address map might be partially populated, with the possibility of fully implemented, partially implemented and unimplemented registers throughout that address space.

The following behaviors are permitted for access to an unimplemented register or an unimplemented register bit in the System MMU configuration space:

- Access to an unimplemented configuration register can be RAZ/WI, or can result in a Configuration access fault. Such behavior is also permitted for a Non-secure access to a configuration resource reserved by Secure software.
- Access to an unimplemented bit in a configuration register where at least one bit is implemented can behave as RAZ/WI.

A System MMU implementation can restrict access to some or all of the configuration addresses based on the privilege of the access, with the following behaviors permitted where the transaction does not have sufficient access permissions:

- treat the access as RAZ/WI
- raise a Configuration access fault.

Unless explicitly specified in relation to a particular configuration address or range of addresses, it is IMPLEMENTATION DEFINED as to which of the permitted behaviors occurs.

If an implementation generates a Configuration access fault in response to Non-secure software accessing a Secure register such as SMMU_SCR1, or a Secure resource such as a context bank reserved for Secure software, the fault is reported to SMMU_SGFSR. The fault also:

- raises an interrupt, if the SMMU_sCR0.GCFGFIE bit is set to 1
- returns an abort, if the SMMU_sCR0.GCFGFRE bit is set to 1.

Secure software can read the SMMU_SGFSYNR0.NSSTATE bit to determine whether a recorded Configuration access fault was generated by a Non-secure transaction.

When the SMMU_SCR1.GASRAE bit is set to 1, the global SMMU address space is accessible only by Secure configuration memory accesses, meaning that all Configuration access faults generated to global SMMU address space registers are also reported to SMMU_SGFSR. This permits Secure software to detect potential security violations by Non-secure software.

Note

This behavior means that Non-secure software can inject faults into SMMU_SGFSR, possibly preventing Secure software from detecting or managing other faults.

3.10 External faults

A fault encountered outside the System MMU can be reported to the System MMU across the system interconnect.

An external fault encountered in an instruction read, a data read or a data write can be reported to the System MMU, which might record the fault synchronously or asynchronously.

External fault support in a System MMU implementation is only required when an external fault is returned in response to a read issued as part of a translation table walk:

- If the translation table walk is part of processing a client access, the System MMU records the external fault in the appropriate fault status register.
- If the translation table walk is part of processing an address translation operation initiated by a configuration access, the fault status is generally captured in SMMU_sGPAR or SMMU_CBn_PAR. An exception to this is where an external fault is encountered during a stage 2 translation table walk that is performed as part of processing an address translation operation initiated in a Stage 1 Translation context bank. In this case, stage 2 SMMU_CB*n*_FSR and related registers might capture a fault status. Depending on how the fault is serviced, the corresponding Stage 1 Translation context bank SMMU_CBn_PAR captures an *Address Translation Operation Terminated* (ATOT) status.

For external faults reported to the System MMU in all other cases, it is IMPLEMENTATION DEFINED which external faults, if any, the System MMU records.

An external abort reported by the System MMU selects a fault status register group and updates the fault status register in this group with a fault status code. A synchronously recorded external abort selects the fault status register group corresponding to the System MMU resources that processed the faulty transaction. If the transaction bypassed translation, the global fault status group is used. If the transaction was processed with one or more Translation context banks, the fault status group belonging to the Translation context bank is used.

For nested translation, the Stage 2 Translation context bank is always used.

For an asynchronously recorded external abort, the fault status register group selected is IMPLEMENTATION DEFINED. The selected register group can be a fault status register group as specified for synchronous reporting or the global fault status register group.

A synchronously recorded external abort updates the fault status register and the fault address and fault syndrome registers. The fault status register is updated to indicate that an external fault has occurred. The fault address and fault syndrome registers are updated with syndrome information about the transaction that caused the fault.

An asynchronously recorded external abort updates the fault status register to indicate that an external fault has occurred. The fault address and syndrome registers are UNKNOWN.

3.10.1 Interaction with the Security Extensions

In an implementation that includes the Security Extensions, an external fault recorded in the global fault status group can update either SMMU_GFSR or SMMU_SGFSR, depending on the security state of the transaction that caused the external fault. See *SMMU_sGFSR, Global Fault Status Register on page 10-131*.

For an asynchronously recorded external abort in the global fault status group, all external faults are permitted to be recorded in SMMU_GFSR. Secure software is provided with an override, SMMU_SCR1.GEFRO, which when set, causes all such faults to instead be recorded in SMMU_SGFSR.

3.11 Reporting exclusive access transactions

Depending on the bus system used, an implementation might support exclusive access transactions, in which case the possible return responses indicate that the transaction has:

- succeeded
- failed
- aborted.

In the SMMU, reporting means specifically reporting an exclusive access transaction as aborting. When the SMMU is configured so that aborts are not reported, that is, when either the SMMU_sCR0.GFRE bit or the SMMU_CBn_SCTLR.CFRE bit is set to 0, ARM recommends that an implementation reports that the transaction has failed.

Chapter 4
Address Translation Commands

This chapter describes address translation commands. It contains the following sections:

4.1 About address translation commands

Software-accessible commands translate the following types of address using a specified Translation context bank:

- stage 1 translation from a virtual address to an intermediate physical address
- single-step translation from a virtual address to a physical address.

The address translation commands are in the global address space and the Stage 1 Translation context bank address space. They are equivalent to the commands in the processor architecture, with some minor differences in the usage model and in the fault handling.

The remainder of this chapter describes the address translation commands in detail.

4.2 Address translation commands in a Stage 1 translation context

Table 4-1 shows the commands that are available in a Stage 1 Translation context bank.

Table 4-1 Stage 1 address translation commands

Command	Functionality
SMMU_CBn_ATS1UR	Stage 1 unprivileged read
SMMU_CBn_ATS1UW	Stage 1 unprivileged write
SMMU_CBn_ATS1PR	Stage 1 privileged read
SMMU_CBn_ATS1PW	Stage 1 privileged write

To begin translation, software writes the address that is to be translated, to the required command address. The stage 1 translation tables give the result of a successful translation. This result is an intermediate physical address.

4.2.1 Usage model

The address translation commands are used with the following registers in the same Translation context bank:

- SMMU_CBn_PAR
- SMMU_CBn_ATSR.

If the operation invoked by the command succeeds, SMMU_CBn_PAR holds the translated address.

SMMU_CBn_ATSR tracks the progress of an address translation operation. SMMU_CBn_ATSR.ACTIVE is set to 1 when a new address translation operation starts, and remains set at this value until the previously requested operation completes. UNPREDICTABLE behavior arises if an address translation operation is requested in the same Translation context bank before the previously requested address translation operation completes.

4.2.2 Fault handling

If an address translation operation fails, SMMU_CBn_PAR generally captures a fault code. See also *Fault handling within nested translation operations initiated in a context bank*.

Note

Address translation operations in a stage 1 translation context operate independently from the translation of client transactions within that stage 1 translation context bank. A fault in a stage 1 context bank, which is indicated by SMMU_CBn_FSR having a non-zero value, does not restrict address translation operations.

4.2.3 Nested translation effect

In an implementation that supports nested translation, if a Stage 1 Translation context bank is associated with a Stage 2 Translation context bank, the stage 1 address translation operation locates the stage 1 translation tables necessary to perform the command using the translations specified by the Stage 2 Translation context bank.

The result of a successful operation is the address specified by the Stage 1 Translation context bank. This is the intermediate physical address, specific to nested translation.

4.2.4 Fault handling within nested translation operations initiated in a context bank

This section applies to address translation operations initiated in a context bank. See *Fault handling on page 4-69* for information about fault handling for global address translations.

If an address translation operation is initiated in a Stage 1 Translation context bank that is associated with a Stage 2 Translation context bank, the handling of a fault that is encountered when processing an address translation operation is modified for the following conditions:

- an external abort
- a fault during stage 2 translation required by a stage 1 initiated address translation.

Depending on the nature of the fault, it is recorded either in the stage 2 context bank or in SMMU_sGFSR. The address translation operation is either suspended or terminated.

Handling faults that are recorded in a stage 2 context bank

This section applies to address translation operations initiated in a context bank. See *Fault handling on page 4-69* for information about fault handling for global address translations.

A fault encountered during an address translation operation initiated in a Stage 1 Translation context bank is recorded in a stage 2 context bank if it is a result of any of the following conditions:

- an external abort
- a fault during stage 2 translation required by a stage 1 initiated address translation, including:
  - a stage 2 permission fault
  - a stage 2 translation fault
  - an external abort on a stage 2 translation table walk.

In such cases, the address translation operation is either suspended or terminated.

If the fault is recorded, that is, if the value in the stage 2 SMMU_CBn_FSR register is 0 prior to the fault, then:

- SMMU_CBn_FSR captures the appropriate fault status
- the stage 2 SMMU_CBn_FAR register captures the input address to the address translation operation

  Note

  This is the virtual address, as would be the case for a nested client transaction faulting in stage 2.

- the stage 2 SMMU_CBn_FSYNR0 register captures other syndrome information about the fault, including:
  - the ATOF bit, to indicate that the fault is a result of a stage 1 address translation operation
  - the S1PTWF bit, to indicate that the fault is related to a stage 1 translation table walk
  - the S1CBNDX bit, which records the stage 1 context bank.
- the value in SMMU_CBFRSYNRAn is UNKNOWN, because no Stream ID or SSD_Index can be associated with the address translation operation.

If the fault is not recorded, that is, if the value in the stage 2 SMMU_CBn_FSR register is not 0 prior to the fault, the stage 2 SMMU_CBn_FSR.MULTI bit is set to 1.

If the fault is recorded and stalling is permitted in stage 2 then the address translation operation is stalled in the same way as a client transaction, and:

- the stage 1 SMMU_CBn_ATSR.ACTIVE register bit is not reset, indicating that the operation is active
- the stage 1 SMMU_CBn_PAR register value remains unknown
- the stage 2 SMMU_CBn_RESUME command can be used to resume the operation in stage 2

If the stage 2 context is configured to use the Terminate fault model, the System MMU terminates the address translation operation, as follows:

- the stage 1 SMMU_CBn_ATSR.ACTIVE field resets
- the stage 1 SMMU_CBn_PAR register updates with the *Address Translation Operation Terminated* (ATOT) fault code, regardless of the value of the stage 2 SMMU_CBn_SCTLR.CFRE bit.

Note

If stage 2 is already faulting and the stage 2 SMMU_CBn_SCTLR.HUPCF bit is set to 0 so that the bank terminates subsequent transactions, then the address translation operation is terminated regardless of whether the operation might have completed successfully.

Handling faults that are recorded in the Global Fault Status Registers

This category of fault handling applies to faults encountered during a stage 2 translation operation that is required by an address translation operation initiated in a Stage 1 context bank. These include:

- stage 2 invalid context faults
- stage 2 unimplemented context faults
- stage 2 invalid context interrupt faults.

Such faults are never stalled as there is no stage 2 to configure the stall model. The SMMU attempts to record these faults in SMMU_sGFSR

If the fault is recorded, that is, if the value in SMMU_sGFSR is 0 prior to the fault:

- SMMU_sGFSR captures the appropriate fault status
- in SMMU_sGFSYNR0, the following bits are set to 1:
  - Nested
  - NSSTATE
  - NSATTR
  - ATS
- in SMMU_sGFSYNR0, the value of the following bits is UNKNOWN:
  - WNR
  - PNU
  - IND.
- the value of the following registers is UNKNOWN:
  - SMMU_sGFAR
  - SMMU_sGFSYNR1
- SMMU_sGFSYNR2 is updated with an IMPLEMENTATION DEFINED value.

Note

These address translation operations are not recoverable, because these faults are caused by the hypervisor failing to configure the SMMU correctly. The UNKNOWN fields cannot be used for diagnostic purposes.

If the fault is not recorded, that is, if the value in SMMU_sGFSR is not 0 prior to the fault:

- the SMMU_sGFSR.MULTI bit is set to 1 and the other bits in SMMU_sGFSR are not updated
- the following registers are not updated:
  - SMMU_sGFAR
  - SMMU_sGFSYNR0
  - SMMU_sGFSYNR1
  - SMMU_sGFSYNR2
- the address translation operation is terminated, meaning that:
  - the stage 1 SMMU_CBn_ATSR.ACTIVE bit is reset
  - the stage 1 SMMU_CBn_PAR register is updated with the ATOT fault code.

4.3 Address translation commands in the global address space

Table 4-2 shows software-accessible commands in the global address space.

Table 4-2 Global address translation commands

Command	Functionality
SMMU_sGATS1UR	Stage 1 unprivileged read
SMMU_sGATS1UW	Stage 1 unprivileged write
SMMU_sGATS1PR	Stage 1 privileged read
SMMU_sGATS1PW	Stage 1 privileged write
SMMU_sGATS12UR	Stage 1 and 2 unprivileged read
SMMU_sGATS12UW	Stage 1 and 2 unprivileged write
SMMU_sGATS12PR	Stage 1 and 2 privileged read
SMMU_sGATS12PW	Stage 1 and 2 privileged write

To begin translation, software writes to the required command address. All of the commands require an input address and a Stage 1 Translation context bank index as arguments.

If a command receives as its input argument, an invalid index from the software, an Unimplemented context bank fault is recorded by the System MMU in the global context bank. An example of an invalid index is an index that corresponds to either:

- a non-existent Translation context bank
- a Translation context bank configured in the format of a Stage 2 Translation context bank.

A Secure command, for example SMMU_SGATS1UR, accepts an index of either a Secure or a Non-secure Stage 1 Translation context bank. A Non-secure command, for example SMMU_GATS1UR, accepts an index of a Non-secure Translation context bank.

See the following sections for more information:

- *Usage model*
- *Fault handling on page 4-69*
- *Nested translation effect on page 4-69*
- *Interaction with the Security Extensions on page 4-69*.

4.3.1 Usage model

Address translation commands are used with the following registers in the same Translation context bank:

- SMMU_sGPAR
- SMMU_sGATSR.

If the operation is successful, the corresponding SMMU_sGPAR register holds the translated address.

SMMU_sGATSR tracks translation progress. When an operation invoked by an address translation command starts, SMMU_sGATSR.ACTIVE is set to 1, and remains set at 1 until the previously requested command completes.

If software invokes an address translation command before the previously requested command completes, the resulting behavior is UNPREDICTABLE.

Secure address translation operations work in combination with SMMU_SGPAR and SMMU_SGATSR. Non-secure address translation operations work in combination with SMMU_GPAR and SMMU_GATSR.

After using any of these commands, software must read SMMU_sGATSR.ACTIVE to determine whether:

- SMMU_sGPAR has been updated to show the outcome of the last requested address translation operation

- a new address translation operation can be initiated.

4.3.2 Fault handling

If a global translation fails, the corresponding SMMU_sGPAR register captures a fault code. The handling of faults arising from global address translation operations is more simple than that of Translation context bank address translation operations, because global address translation operations do not require any consideration for the interaction between a hypervisor and an operating system.

Global address translation operations that are initiated in global register space are not subject to stalls. For these operations, the SMMU reports faults to SMMU_sGPAR rather than the stage 1 or stage 2 context resources, regardless of whether:

- the translation is nested
- the context banks identified are configured to stall or terminate
- the context bank is already faulting.

In addition, these operations are not queued behind client transactions and other translation requests in a stalling context bank that is recording a fault.

If a fault is encountered during a local address translation operation, the SMMU_sGPAR records the type of fault and the address translation stage in which it occurred.

4.3.3 Nested translation effect

In an implementation that supports nested translation, if a Stage 1 Translation context bank is associated with a Stage 2 Translation context bank, the stage 1 address translation operation locates the stage 1 translation tables necessary to perform the command using the translations specified by the Stage 2 Translation context bank.

If the stage 1 Translation context bank is associated with a stage 2 fault context, the address translation operation fails and an Invalid context fault occurs.

4.3.4 Interaction with the Security Extensions

In an implementation that includes the Security Extensions, equivalent functionality provides Secure configuration accesses in the form of SMMU_SGPAR, SMMU_SGATSR and SMMU_SGATxx operations. See SMMU_sGPAR, SMMU_sGATSR and Table 4-2 on page 4-68 for more information.

The Secure global address translation operations operate independently to the Non-secure global address translation operations. A Secure command accepts an index of either a Secure or a Non-secure Translation context bank. A Non-secure command accepts an index of a Non-secure Translation context bank. Aliases in the global address space give Secure software access to the Non-secure address translation commands and registers.

Chapter 5
Coherency Issues

This chapter describes coherency issues. It contains the following sections:

5.1 Atomic update of state

The System MMU architecture does not provide any special support for the atomic update of System MMU state.

Controlling software is responsible for coordinating the update of System MMU state so that a non-atomic update does not interfere with concurrent accesses. Before a write, controlling software must ensure that no transaction that might be affected by a change of SMMU state is in progress. A transaction that is unaffected by a change of state is permitted to continue concurrently with the change in System MMU state.

The following areas might require special care:

- the context mapping group registers
- the stage 1 SMMU_CBn_TTBRm and stage 2 SMMU_CBn_TTBR0 registers
- the *Address Space IDentifier* (ASID) registers
- the *Virtual Machine ID* (VMID) registers
- the action of clearing fault status values.

5.2 Translation table walk coherency

The diverse range of system topologies that a System MMU might be deployed in means it is not always possible to guarantee a coherent translation table walk. Therefore, it is IMPLEMENTATION DEFINED whether a System MMU translation table walk is coherent.

SMMU_IDR0.CTTW indicates whether support for coherent translation table walks is implemented.

For a system that is being designed to support mainstream platform operating systems, ARM strongly recommends the inclusion of coherent translation table walks.

Under the ARMv7 Multiprocessing Extensions, an ARM processor translation table walk must access its own data or unified cache, or the data or unified cache of another agent participating in the coherency protocol, according to the shareability attributes described in the stage 1 SMMU_CBn_TTBRm or stage 2 SMMU_CBn_TTBR0 register. The shareability attributes described in these registers must be consistent with the shareability attributes for the translation tables.

5.3 Broadcast TLB maintenance operations

The diverse range of system topologies that a System MMU might be deployed in means it is not always possible to guarantee support for broadcast TLB maintenance operations. Therefore, it is IMPLEMENTATION DEFINED whether a System MMU supports broadcast TLB maintenance operations.

SMMU_IDR0.BTM indicates support for broadcast TLB maintenance operations.

For a system that is being designed to include mainstream platform operating systems, ARM strongly recommends the provision of support for broadcast TLB maintenance operations.

5.3.1 Private VMID namespace

In a system where the system software has a different VMID namespace between the processor and the System MMU, software can hint to the System MMU that any broadcast TLB maintenance operation specifying a VMID is to be ignored. SMMU_sCR0.VMIDPNE provides this hint.

For example, a broadcast Non-secure SMMU_CBn_TLBIVA operation that specifies a VMID is not required to affect the TLB entries. However, a broadcast SMMU_TLBIALLNSNH must affect all TLB entries that are tagged Non-secure and non-hypervisor.

Note

This hint has no effect on Secure broadcast TLB Invalidate operations because they have no associated VMID.

5.3.2 Private ASID namespace

In a system where the system software has a different ASID namespace between the processor and the System MMU, software can hint to the System MMU that any broadcast TLB maintenance operation specifying an ASID is to be ignored. SMMU_CBn_SCTLR.ASIDPNE provides this hint.

5.4 Memory-mapped TLB maintenance operations

A set of memory-mapped TLB maintenance operations explicitly invalidates translation table entries held in a TLB by the System MMU implementation. They are provided in the following locations of the System MMU address space:

- The Translation context bank address space. See:
  - *SMMU_CBn_TLBIALL, TLB Invalidate All on page 15-215*
  - *SMMU_CBn_TLBIASID, TLB Invalidate by ASID on page 15-216*
  - *SMMU_CBn_TLBIVA, Invalidate TLB by VA on page 15-216*
  - *SMMU_CBn_TLBIVAA, TLB Invalidate by VA All ASID on page 15-217*
  - *SMMU_CBn_TLBIVAAL, TLB Invalidate by VA, All ASID, Last level on page 15-218*
  - *SMMU_CBn_TLBIVAL, TLB Invalidate by VA, Last level on page 15-218*
  - *SMMU_CBn_TLBSTATUS, TLB Status on page 15-219*
  - *SMMU_CBn_TLBSYNC, TLB Synchronize Invalidate on page 15-220*.
- The global address space. See:
  - *SMMU_sTLBGSTATUS, Global TLB Status register on page 10-138*
  - *SMMU_sTLBGSYNC, Global Synchronize TLB Invalidate on page 10-139*
  - *SMMU_STLBIALL, TLB Invalidate All on page 10-149*
  - *SMMU_TLBIALLH, TLB Invalidate All Hyp on page 10-150*
  - *SMMU_TLBIALLNSNH, TLB Invalidate All Non-Secure Non-Hyp on page 10-150*
  - *SMMU_TLBIVMID, TLB Invalidate by VMID on page 10-151*
  - *SMMU_TLBIVAH, Invalidate Hyp TLB by VA on page 10-150*.

Operations in the Translation context bank address space are for the maintenance of translation tables associated with a particular translation context. Because of the location of the operations in the Translation context bank address space, a Guest OS can directly maintain its own translations.

Operations in the global address space are for supervisory code that might have to perform maintenance operations and confirm completion of TLB maintenance operations globally across the System MMU instance.

A write to any of the following locations is an invalidation request. The SMMU_CBn_TLBSYNC command and the SMMU_CBn_TLBSTATUS register manage the completion of each operation:

- SMMU_CBn_TLBIALL
- SMMU_CBn_TLBIASID
- SMMU_CBn_TLBIVA
- SMMU_CBn_TLBIVAA
- SMMU_CBn_TLBIVAAL
- SMMU_CBn_TLBIVAL.

A write to SMMU_CBn_TLBSYNC is a synchronization request. SMMU_CBn_TLBSTATUS indicates the completion of the request.

A write to any of the following locations is a global invalidation request:

- SMMU_STLBIALL
- SMMU_TLBIALLH
- SMMU_TLBIALLNSNH
- SMMU_TLBIVMID.

A write to SMMU_sTLBGSYNC is a synchronization request As a minimum, the synchronization operation applies to the specified security state, and includes all TLB Invalidate operations initiated in context banks associated with that security state.

SMMU_sTLBGSTATUS indicates completion of all the TLB invalidate operations initiated before the most recent write to SMMU_sTLBGSYNC.

5.4.1 TLB maintenance operation processing

Memory-mapped TLB maintenance operations have a post and synchronize model, where a TLB Invalidate operation is requested by writing to the appropriate command, and completion is confirmed by a subsequent synchronization operation.

An accepted TLB Invalidate operation is not complete until:

- every translation held in a TLB that is a target of a TLB Invalidate operation has been discarded
- every transaction already in progress that has used the translations held in the TLB has been globally observed.

A System MMU must accept an unbounded number of memory-mapped TLB maintenance operations without relying on the forward progress of client transactions.

Software can use a SYNC operation to determine that a memory-mapped TLB maintenance operation is complete. A SYNC operation accepted in a Translation context bank only ensures the completion of a TLB maintenance operation accepted in that Translation context bank. A SYNC operation accepted in the global address space ensures the completion of a TLB maintenance operation accepted in either the global address space or in any Translation context bank.

POST

The outline assembly language source code for posting a new TLB Invalidate operation is as follows, assuming the operation is to invalidate by virtual address:

```
MOV     R0,#VA
MOV     R1,#SMMU_CBn_TLBIVA
STR     R0,[R1]
```

SYNC

The outline assembly language source code for ensuring the completion of one or more posted TLB Invalidate operations is as follows:

```
    MOV     R0,#SMMU_CBn_TLBSYNC
    MOV     R1,#SMMU_CBn_TLBSTATUS
    STR     R0,[R0] ; Initiate TLB SYNC
Loop:
    LDR     R0,[R1]
    ANDS    R0,R0, #1 ; TLB SYNC ACTIVE STATUS
    BNE     Loop
    DSB
```

5.4.2 Thread safety

Memory-mapped TLB maintenance operations do not provide atomic behavior. However, the state necessary for initiating and tracking the completion of a TLB maintenance operation is duplicated so that multiple threads can work independently:

- in an implementation that includes the Security Extensions, the global TLB maintenance operation state is banked for security
- the Translation context bank TLB maintenance operation state is provided per Translation context bank.

Software arbitration is required if there is the potential for multiple threads of activity to use the same TLB maintenance operation state concurrently. For example, a software lock is required if it is possible for multiple threads to attempt to perform TLB maintenance operations in a single Translation context bank.

Chapter 6
Debug Support

This chapter outlines debug support. It contains the following section:

- *TLB visibility on page 6-78*.

6.1 TLB visibility

The System MMU architecture does not require the provision of debug support features. However, ARM strongly recommends that an implementation provides a mechanism to read the content of any TLB structure in the implementation. The method by which the System MMU provides this feature is IMPLEMENTATION DEFINED. Space is reserved in the global register map to provide access to such a function. See Chapter 10 *System MMU Global Register Space 0* for more information.

If an implementation includes the Security Extensions, a Non-secure debug agent must not be able to read any data relating to Secure transaction handling.

Chapter 7
System MMU Performance Monitors Extension

This chapter describes the System MMU Performance Monitors Extension. It contains the following sections:

7.1 About the System MMU Performance Monitors Extension

The System MMU Performance Monitors Extension is an OPTIONAL memory-mapped extension. If required, the extension can be implemented as a CoreSight component, by including the CoreSight Component and Peripheral ID registers defined in the *CoreSight Architecture Specification*. Whether a System MMU includes the Performance Monitors Extension is IMPLEMENTATION DEFINED. If not supported, the register map corresponding to the Performance Monitors registers is UNK/SBZP.

If implemented, the System MMU Performance Monitors Extension provides event counter resources and event filtering based on a translation context or a StreamID. Event counter resources are revealed in the address map of a Translation context bank. The configuration of these resources permits a Guest OS to use them without the intervention of a hypervisor.

7.2 The register map

The System MMU Performance Monitors Extension supports a memory-mapped register interface in the System MMU global address space. This interface occupies a *PAGESIZE* region starting at an offset of (3 × *PAGESIZE*) from the System MMU base address. See Chapter 9 *System MMU Address Space* and *PAGESIZE and NUMPAGENDXB on page 9-98* for more information.

The System MMU Performance Monitors Extension also provides the capability to reveal a group of performance monitoring resources in the register map of a Translation context bank. See *Translation context bank assignment on page 7-86* for more information.

7.3 Event classes

Table 7-1 shows the event classes of the System MMU Performance Monitors Extension.

Table 7-1 System MMU performance monitoring events

Category	Event Number	Description
Cycle	0x00	Cycle count, occurs every System MMU clock cycle
	0x01	Cycle count divided by 64 event, occurs every 64th System MMU clock cycle
TLB	0x08	TLB Refill, occurs when a System MMU refills a TLB to load a translation
	0x09	TLB Refill Read
	0x0A	TLB Refill Write
Access	0x10	Access, occurs when a System MMU processes a new transaction
	0x11	Access Read
	0x12	Access Write

It is IMPLEMENTATION DEFINED which event classes are included in an implementation. See *PMCEID0, Performance Monitors Common Event Identifier 0 register on page 13-171* for more information.

All unused encodings are reserved:

- unused encodings in the range 0x00-to 0x7F are reserved for future architectural event classes
- unused encodings in the range 0x80-to 0xFF are reserved for IMPLEMENTATION DEFINED event classes.

7.4 StreamID groups

The System MMU Performance Monitors Extension includes the concept of a StreamID group. A StreamID group is a set of StreamIDs. Transactions with a StreamID that is a member of a StreamID group are associated with that group. The number of StreamID groups, and the mapping of StreamIDs to StreamID groups, is IMPLEMENTATION DEFINED.

Event counters are affiliated with a StreamID group. An event counter can only count events caused by the processing of transactions associated with that group.

It is permissible to define a number of StreamID groups with potentially overlapping membership. A global StreamID group can be defined to contain all StreamIDs as members. ARM suggests StreamID group 0 be reserved for this purpose.

The concept of StreamID groups caters for distributed systems where a remote TLB might service only a subset of client devices, therefore only having visibility of a subset of transactions processed by the System MMU. The StreamID group definition makes counting events limited to such a subset of transactions permissible.

Event counters are affiliated with a StreamID group on a fixed basis. There is no mechanism to change the relationship. The affiliation with a StreamID is made on a per-counter group basis.

7.5 Counter groups

The System MMU Performance Monitors Extension provides:

- Translation context event filtering
- StreamID event filtering
- Translation context bank assignment.

These mechanisms are configured by controls that operate on a group of counters known as a Counter group.

The number of event counters implemented, and the number of Counter Groups, is IMPLEMENTATION DEFINED:

- PMCFGR.N indicates the total number of implemented event counters
- PMCFGR.NCG indicates the number of Counter groups
- For each Counter group, PMCGCRn.CGNC indicates the IMPLEMENTATION DEFINED number of event counters associated with Counter group *n*.

The System MMU architecture permits a maximum of:

- 256 implemented event counters in total
- 256 Counter groups
- 15 counters in a Counter group.

Event counters are arranged by their Counter group, with counters associated with Counter group 0 occurring first in the event counter address map.

PMCGCRn and PMCGSMRn configure the following features by Counter group:

- Translation context event filtering
- StreamID event filtering
- Translation context bank assignment features.

7.6 Event filtering

A Counter group counts events on a global, translation context, or StreamID basis:

- If configured to count all events, it considers all transactions associated with the StreamID group that the Counter group is affiliated with.
- If configured to filter events on a translation context basis, it only considers transactions processed by the System MMU translation context designated by PMCGCRn.NDX, and limited by the scope of the StreamID group that the Counter group is affiliated with.
- If configured to filter events on a StreamID basis, it only considers transactions that match the ID and the mask designated by PMCGSMRn.ID and PMCGSMRn.MASK, and limited by the scope of the StreamID group that the Counter group is affiliated with. An implementation must provide the same number of PMCGSMRn.ID and PMCGSMRn.MASK bits for every implemented PMCGSMRn register.

PMCGCRn.TCEFCFG selects the type of filtering on which the event counting is based.

7.7 Translation context bank assignment

The System MMU Performance Monitors Extension provides the capability to reveal a Counter group in the register map of a Translation context bank. A hypervisor can use this functionality to give a Guest OS direct access to performance monitoring resources, to profile the behavior of any device associated with the translation context that the Guest OS uses.

A Counter group and its related registers can be revealed in a Translation context bank register map using the PMCGCRn registers associated with that Counter group:

- PMCGCRn.CBAEN enables Translation context bank assignment
- if enabled by PMCGCRn.CBAEN, PMCGCRn.NDX specifies the Translation context bank to assign the Counter group to.

Only one Counter group can be revealed in a Translation context bank. Behavior is UNPREDICTABLE if multiple PMCGCRn registers specify the same Translation context bank.

Translation context bank assignment can only be enabled after the following event filtering modes of operation are configured:

- Translation context event filtering
- StreamID event filtering.

Behavior is UNPREDICTABLE if Translation context bank assignment is enabled when global event monitoring and filtering is enabled.

PMCGCRn and PMCGSMRn are not revealed in the Translation context bank register map. This would cause a virtualization hole.

When a Counter group is revealed in a Translation context bank, one of the registers revealed is an SMMU_CBn_PMCR register associated with that group. This is a Banked register.

7.7.1 Translation context register map

The Translation context bank register map occupies an address space defined by *PAGESIZE*. When a Counter group is revealed in a Translation context bank register map, a more compressed register layout balances the use of available space versus future expansion. The layout of performance monitor controls in a Translation context bank register map places an upper limit on the number of event counters that can exist in a Counter group.

A Counter group that is revealed in a Translation context register map is self-contained. The event counters revealed in the Translation context register map are numbered beginning at 0. The Performance Monitor ID and configuration registers in the Translation context register map indicate the number of event counters contained in the Counter group, not the total number implemented in the System MMU.

7.7.2 SMMU_CB*n*_PMCR banking

PMCR only operates on event counters of Counter groups that are not revealed in a Translation context bank.

A separate register, SMMU_CBn_PMCR, controls event counters in a Counter group that is revealed in a Translation context bank. This register is revealed as part of the designated Translation context bank. For state save and restore purposes, the active RW bits of this Banked register are available in the PMCGCRn register associated with that Counter group.

7.7.3 Counter group absent

If no Counter group is revealed in a Translation context bank, default read-only RAZ/WI values are presented at the Performance Monitors register locations in that Translation context bank.

7.8 Event counter overflow interrupt

The System MMU Performance Monitors Extension provides the capability to assert an interrupt in the event of an event counter overflowing. PMINTENSETx and PMINTENCLRx enable and disable interrupt assertion on a per event counter basis.

The interrupt that is asserted depends on the Counter group that the event counter is a member of. One interrupt output is provided per Counter group.

These interrupt outputs are separate from the global and translation context interrupt outputs, because different software modules are generally involved in performance monitoring.

7.9 Interaction with the Security Extensions

There are no facilities for Secure software to reserve performance monitoring resources. If required, this can be achieved through a collaborative software protocol between the Secure and the Non-secure software domains.

By default, an event that results from processing a Secure transaction does not contribute to performance monitor counting. SMMU_SCR1.SPMEN enables the counting of such events, as does the external Secure PL1 Non-invasive debug enable input signal, **SPNIDEN**.

Chapter 8
Security Extensions

This chapter describes the relationship between the System MMU architecture and the OPTIONAL Security Extensions. It contains the following sections:

8.1 Sharing resources between Secure and Non-secure domains

In an implementation that includes the Security Extensions, the resources of a System MMU are shared between Secure and Non-secure domains. For information about the Security Extensions, see the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition*.

The System MMU architecture permits inclusion of the Security Extensions to be an IMPLEMENTATION DEFINED choice. SMMU_IDR0.SES indicates whether the Security Extensions are included.

8.2 Excluding the Security Extensions

A System MMU implementation might exclude the Security Extensions. However, this does not mean that a transaction from a Secure device cannot arrive at the System MMU. If a System MMU does not include the Security Extensions, and is expected to receive and process any transaction from a Secure device, security state determination must ensure that this transaction bypasses all subsequent System MMU transaction processing.

If SMMU_IDR0.SES is 0:

- the System MMU implementation does not translate transactions from Secure devices
- Stream match register groups, Translation context banks and interrupts are not reserved
- the Secure Banked control and status registers are absent.

Note

Regardless of whether a System MMU implementation excludes the Security Extensions, the introduction of a System MMU must not create any type of security loophole.

8.3 Including the Security Extensions

If SMMU_IDR0.SES is 1, Secure software might arrange for a System MMU to process and translate transactions from one or more Secure devices.

8.3.1 Translation restrictions

In a System MMU implementation that includes the Security Extensions, the following restrictions apply to a transaction from a Secure device:

- stage 2 nested translation is not permitted
- a transaction from a Secure device must only be translated by a stage 1 context that is reserved by Secure software.

8.3.2 Resource reservation

In a System MMU implementation that includes the Security Extensions, a number of resources are shared between Secure and Non-secure domains. For some of these resources, Secure software might reserve some or all of the resource for the sole use of the Secure software. Such shared resources include:

- Stream mapping register groups. See SMMU_SCR1.NSNUMSMRGO.
- Context interrupts. See SMMU_SCR1.NSNUMIRPTO.
- Translation context banks. See SMMU_SCR1.NSNUMCBO.

The SMMU_IDR*x* registers take into account the number of registers reserved by Secure software when reporting the number of resources. See *SMMU_IDR0-7, Identification registers on page 10-116*.

If Secure software reserves a shared System MMU resource, one of the following generally applies:

- the reservation occurs at Secure system boot time and is static for the duration of system uptime
- a software interface between Secure and Non-secure domains is implemented that supports the dynamic partitioning of System MMU resources.

As a consequence of the restrictions specified in *Translation restrictions*, the following conditions apply:

- A transaction that is determined to be Secure by security state determination must only match a Stream mapping register group that is reserved by Secure software.
- A Stream mapping register group that is reserved by Secure software must only specify:
  - a Translation context bank reserved by Secure software
  - Bypass mode
  - the fault context.
- The SMMU_CBARn register associated with a Translation context bank reserved by Secure software must only specify a Context interrupt that is reserved by Secure software.
- Any Translation context bank reserved by Secure software must be placed above the Translation context bank indicated by SMMU_IDR1.NUMS2CB. This field indicates the last Translation context bank that only supports the stage 2 translation format.

8.3.3 Permitted transaction resource usage

This section specifies which resources a transaction is permitted to interact with, in relation to the Security Extensions.

The Security Extensions in the processor architecture introduce a Secure domain and a Non-secure domain, based on the following fundamental concepts:

- The Secure domain must be able to operate in isolation from the Non-secure domain.

This implies that the Secure domain must have access to registers, instruction memory, and data memory that the Non-secure domain does not have access to. Under the Security Extensions, this is achieved by a combination of dedicated Secure resources and shared resources that the Secure domain acts as a gatekeeper for. In the System MMU architecture, this concept is extended by the ability of Secure software to reserve the following System MMU resources:

— Stream mapping register groups
— Translation context banks
— Context interrupts.

- Secure and Non-secure domains must be able to communicate. Under the Security Extensions this is primarily facilitated by permitting the Secure domain to access Non-secure memory.

Under the processor architecture, the basic model is extended in the following ways:

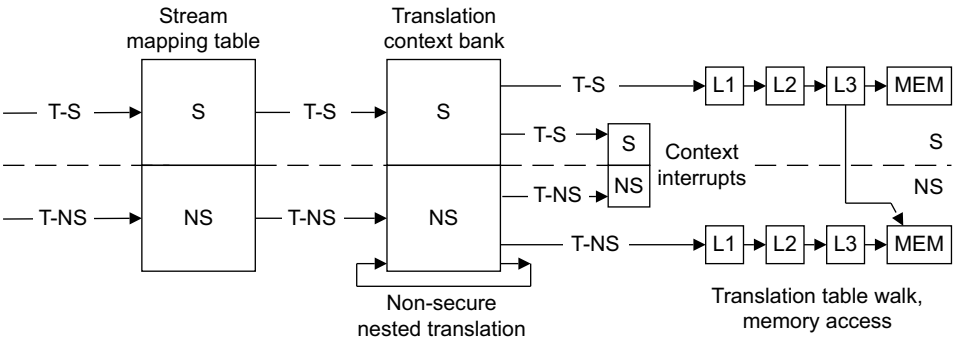
- It is generally useful to give the Secure domain read and write access to any Non-secure domain state in the system.

Note

Read and write access to the Non-secure state is not equivalent to being able to use that state in all cases. For example, Secure software can read from and write to the Non-secure SMMU_CBn_TTBCR registers, but cannot execute a LDR or STR instruction using those registers.

- Inside Non-secure memory, the Secure domain can store translation table mappings that specify translations to Non-secure memory. This reduces the requirements to use Secure memory, which is generally an on-chip, and therefore expensive, resource.

Figure 8-1 shows the relationship between transaction processing and resource usage. The connections shown are mandatory for enabling basic and fundamental operation.



S = Secure resource managed by Secure software
NS = Non-secure resource managed by Non-secure software
T-S = Transaction originating from bus master performing Secure domain operation
T-NS = Transaction originating from bus master performing Non-secure domain operation

Figure 8-1 Basic relationship between processing transactions and using resource

With regard to boundary control between Secure and Non-secure resources:

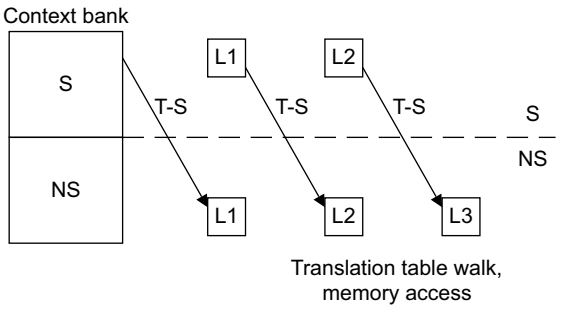
- Secure software controls the boundary between Secure and Non-secure resource.
- In the Stream mapping table, Secure software can claim Stream mapping register groups using SMMU_SCR0.NSNUMSMRGO. See *SMMU_sCR0, Configuration Register 0 on page 10-120*.
- In the Translation context bank space, SMMU_SCR0.NSNUMCBO enables Secure software to adjust the number of Translation context banks visible to Non-secure accesses. This control also has the effect of reserving the SMMU_CBARn and SMMU_CBFRSYNRAn registers associated with the Translation context banks that have been reserved.

- In the Context interrupts space, SMMU_SCR1.NSNUMIRPTO can be configured to reserve some of the interrupts for Secure software.

In addition to the basic relationship between transaction processing and resource usage, the following connections are permitted:

- connections that are not ruled out by security requirements
- connections that have existing equivalent functionality
- connections that are permitted because of other perceived benefits.

Figure 8-2 illustrates these connections.



S, L1, L2 = Secure resource managed by Secure software
NS, L1, L2, L3 = Non-secure resource managed by Non-secure software
T-S = Transaction originating from bus master performing Secure domain operation

Figure 8-2 Permitted relationships between Secure and Non-secure resource usage

Figure 8-1 on page 8-93 and Figure 8-2 show all permitted relationships between Secure and Non-secure resource usage. No other relationships are permitted.

Prohibited relationships

Security requirements mean that the following relationships are not permitted:

- a transaction originating from a bus master operating for the Non-secure domain must never be permitted to map to a transaction stream or translation context in the Secure part of the Stream mapping table
- a transaction associated with the Non-secure part of the Stream mapping table must never be associated with a Secure Translation context bank
- a transaction associated with a Non-secure Translation context bank must never be associated with a Secure Context interrupt
- a transaction associated with a Non-secure Translation context bank must never be associated with a Secure level 1 translation table walk
- a transaction in the Non-secure level 1 part of a translation table walk must never be permitted to enter the Secure level 2 part of a translation table walk
- a transaction in the Non-secure level 2 part of a translation table walk must never be permitted to enter the Secure level 3 part of a translation table walk
- a transaction in the Non-secure level 3 part of a translation table walk must never be associated with target memory managed by Secure software.

In addition, the System MMU architecture prohibits:

- a transaction originating from a bus master operating for the Secure domain must not be permitted to map to a transaction stream or translation context in the Non-secure part of the Stream mapping table

- a transaction associated with the Secure part of the Stream mapping table must not be associated with a Non-secure Translation context bank
- nested translation is not permitted in the Secure part of the Translation context bank
- a transaction associated with a Secure Translation context bank must not be associated with a Non-secure Context interrupt.

Chapter 9
System MMU Address Space

This chapter specifies the address space of a System MMU implementation. It contains the following sections:

9.1 About the System MMU address space

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

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

- the global address space
- the Translation context bank address space.

The global address space is at the bottom of the System MMU address space, *SMMU_BASE*, where *SMMU_BASE* is aligned to (*PAGESIZE* * *NUMPAGE* * 2). The Translation context bank address space is located above the top of the global address space, as shown in Figure 9-1.

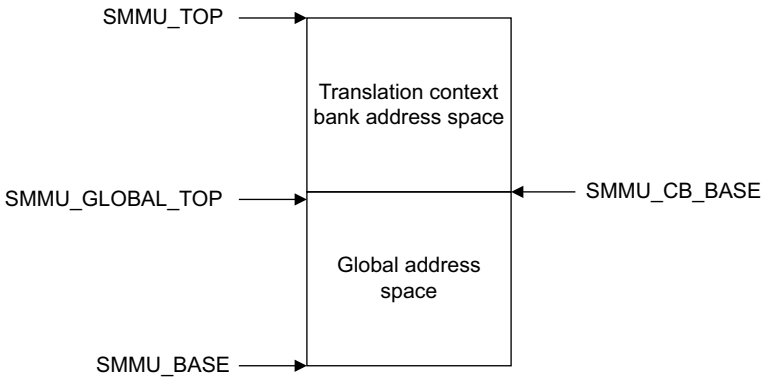


Figure 9-1 System MMU address space

The following register fields indicate the size of the System MMU address space:

- SMMU_IDR1.PAGESIZE
- SMMU_IDR1.NUMPAGENDXB.

9.1.1 *PAGESIZE* and *NUMPAGENDXB*

A System MMU register map arranges state into a number of pages. Each page occupies, and is aligned to, a *PAGESIZE* space in the address map. Such organization permits a hypervisor to permit or deny access to System MMU state on a page-by-page basis.

The System MMU architecture permits an implementation to support either 4KB or 64KB *PAGESIZE* options.

SMMU_IDR1.PAGESIZE specifies the implemented *PAGESIZE*.

NUMPAGE, the number of pages implemented in the global address space or Translation context bank address space is determined from SMMU_IDR1.NUMPAGENDXB, where $NUMPAGE = 2^{(NUMPAGENDXB+1)}$.

9.1.2 Address space calculation

The following byte address calculations give the sizes and base addresses in the System MMU address space:

- $SMMU\_GLOBAL\_SIZE = SMMU\_CB\_SIZE = (NUMPAGE \times PAGESIZE)$
- $SMMU\_TOP = SMMU\_BASE + SMMU\_GLOBAL\_SIZE + SMMU\_CB\_SIZE - 1$
- $SMMU\_GLOBAL\_TOP = SMMU\_BASE + SMMU\_GLOBAL\_SIZE - 1$
- $SMMU\_CB\_BASE = SMMU\_BASE + SMMU\_GLOBAL\_SIZE$
- $SMMU\_CBn\_BASE = SMMU\_CB\_BASE + (n \times PAGESIZE)$.

9.2 The global address space

Table 9-1 shows the System MMU global address space.

Table 9-1 System MMU Global address space

Offset from SMMU_BASE	Description	Notes
0x00000 to ((1 × *PAGESIZE*) − 0x4)	SMMU Global Register Space 0	*PAGESIZE* global address space
(1 × *PAGESIZE)* to ((2 × *PAGESIZE*) − 0x4)	SMMU Global Register Space 1	*PAGESIZE* global address space
(2 × *PAGESIZE)* to ((3 × *PAGESIZE*) − 0x4)	SMMU IMPLEMENTATION DEFINED address space	*PAGESIZE* global address space
(3 × *PAGESIZE)* to ((4 × *PAGESIZE*) − 0x4)	SMMU performance monitoring address space	*PAGESIZE* global address space
(4 × *PAGESIZE*) to ((5 × *PAGESIZE*) − 0x4)	SMMU security state determination address space	*PAGESIZE* global address space, Secure only
(5 × *PAGESIZE)* to *SMMU_GLOBAL_TOP*	Reserved	RAZ/WI

9.2.1 Global address space base addresses

The global address space base addresses defined in Table 9-2 are referred to in subsequent sections.

Table 9-2 Global address space base addresses

Base address name	Description	Value
SMMU_GR0_BASE	Base address of SMMU Global Register Space 0	*SMMU_BASE* + (0 × *PAGESIZE*)
SMMU_GR1_BASE	Base address of SMMU Global Register Space 1	*SMMU_BASE* + (1 × *PAGESIZE*)
SMMU_GID_BASE	Base address of SMMU IMPLEMENTATION DEFINED address space	*SMMU_BASE* + (2 × *PAGESIZE*)
SMMU_PM_BASE	Base address of SMMU performance monitoring address space	*SMMU_BASE* + (3 × *PAGESIZE*)
SMMU_SSD_BASE	Base address of SMMU security state determination address space	*SMMU_BASE* + (4 × *PAGESIZE*)

For more information, see:

- Chapter 10 *System MMU Global Register Space 0*
- Chapter 11 *System MMU Global Register Space 1*
- Chapter 12 *System MMU implementation defined Address Space*
- Chapter 13 *System MMU Performance Monitors Extension Register Map*
- Chapter 14 *The Security State Determination Address Space*.

9.2.2 Interaction with the Security Extensions

In an implementation that includes the Security Extensions, the global address space is generally accessible by both Secure and Non-secure configuration accesses. Some address ranges can only be accessed by Secure configuration accesses.

Several registers are Banked. The security status of a configuration access selects the appropriate resource. Alias registers and commands are provided at distinct addresses to give Secure software access to Non-secure versions of the resources.

Setting SMMU_SCR1.GASRAE to 1 enables a restricted access mode of operation. In this mode, with the possible exception of the IMPLEMENTATION DEFINED address space, all of the global address space and the stage 2 format Translation context banks are accessible by Secure configuration access only. It is IMPLEMENTATION DEFINED whether SMMU_SCR1.GASRAE only permits Secure accesses to the IMPLEMENTATION DEFINED address space.

These restrictions are in addition to any underlying Secure-only resource that might exist.

9.3 The Translation context bank address space

Table 9-3 shows how the System MMU Translation context bank address space is organized, relative to the System MMU context bank base address, *SMMU_CB_BASE*.

Table 9-3 Translation context bank address space

Offset	Description	Notes
`0x00000` to ((1 × *PAGESIZE*) − `0x4`)	Translation context bank 0	*PAGESIZE* per Translation context bank
(1 × *PAGESIZE)* to ((2 × *PAGESIZE*) − `0x4`)	Translation context bank 1	-
(*n* × *PAGESIZE)* to ((*n* + 1 × *PAGESIZE*) − `0x4`)	Translation context bank *n*	-

SMMU_IDR1.NUMCB specifies the IMPLEMENTATION DEFINED number of Translation context banks.

In an implementation that supports nested translation, some Translation context banks must be configured as stage 1 translation contexts, and the remaining Translation context banks as stage 2 translation contexts. Because of the increased state requirements of stage 1 translation contexts, some translation contexts might only support stage 2 translation. SMMU_IDR1.NUMS2CB provides more information about discovering whether an implementation uses such contexts.

Any address above the upper implemented Translation context bank address and below *SMMU_TOP* behaves as RAZ/WI.

Each Translation context bank is defined in terms of its Context Bank Base Address, *SMMU_CBn_BASE*. The base address for Translation context bank *n* is defined as *SMMU_CBn_BASE* = *SMMU_CB_BASE* + (*n* × *PAGESIZE*).

For more information, see:

- Chapter 15 *Stage 1 Translation Context Bank Format*
- Chapter 16 *Stage 2 Translation Context Bank Format*.

Chapter 10
System MMU Global Register Space 0

This chapter specifies System MMU Global Register Space 0. It contains the following sections:

10.1 System MMU Global Register Space 0 register summary

System MMU Global Register Space 0 provides high-level System MMU resource control. The size of this address space is defined by *PAGESIZE*. See *PAGESIZE and NUMPAGENDXB on page 9-98* for more information.

Table 10-1 shows the address of each register relative to the offset from *SMMU_GR0_BASE*.

Table 10-1 System MMU Global Register Space 0

Offset	Name	Type	Description	Notes
0x00000	SMMU_sCR0	RW	*SMMU_sCR0, Configuration Register 0 on page 10-120*	Banked with security
0x00004	SMMU_SCR1	RW	*SMMU_SCR1, Secure Configuration Register 1 on page 10-145*	Secure only
0x00008	SMMU_*s*CR2	RW	*SMMU_sCR2, Configuration Register 2 on page 10-125*	Banked with security
0x0000C	Reserved	-	-	-
0x00010	SMMU_sACR	RW	*SMMU_sACR, Auxiliary Configuration Register on page 10-120*	Banked with security
0x00014-0x0001C	Reserved	-	-	-
0x00020	SMMU_IDR0	RO	*SMMU_IDR0-7, Identification registers on page 10-116*	-
0x00024	SMMU_IDR1			
0x00028-0x0003C	SMMU_IDR2-SMMU_IDR7			
0x00040	SMMU_sGFAR[31:0]	RW	*SMMU_sGFAR, Global Fault Address Register on page 10-130*	Banked with security
0x00044	SMMU_sGFAR[63:32]			
0x00048	SMMU_sGFSR	RW	*SMMU_sGFSR, Global Fault Status Register on page 10-131*	Banked with security
0x0004C	SMMU_sGFSRRESTORE	WO	*SMMU_sGFSRRESTORE, Global Fault Status Restore Register on page 10-132*	Banked with security
0x00050	SMMU_sGFSYNR0	RW	*SMMU_sGFSYNR0, Global Fault Syndrome Register 0 on page 10-133*	Banked with security
0x00054	SMMU_sGFSYNR1	RW	*SMMU_sGFSYNR1, Global Fault Syndrome Register 1 on page 10-134*	Banked with security
0x00058	SMMU_sGFSYNR2	RW	*SMMU_sGFSYNR2, Global Fault Syndrome Register 2 on page 10-135*	Banked with security
0x0005C	Reserved	-	-	-
0x00060	SMMU_STLBIALL	WO	*SMMU_STLBIALL, TLB Invalidate All on page 10-149*	Secure only
0x00064	SMMU_TLBIVMID	WO	*SMMU_TLBIVMID, TLB Invalidate by VMID on page 10-151*	-
0x00068	SMMU_TLBIALLNSNH	WO	*SMMU_TLBIALLNSNH, TLB Invalidate All Non-Secure Non-Hyp on page 10-150*	-

Table 10-1 System MMU Global Register Space 0 (continued)

Offset	Name	Type	Description	Notes
0x0006C	SMMU_TLBIALLH	WO	*SMMU_TLBIALLH, TLB Invalidate All Hyp on page 10-150*	-
0x00070	SMMU_sTLBGSYNC	WO	*SMMU_sTLBGSYNC, Global Synchronize TLB Invalidate on page 10-139*	Banked with security
0x00074	SMMU_sTLBGSTATUS	RO	*SMMU_sTLBGSTATUS, Global TLB Status register on page 10-138*	Banked with security
0x00078	SMMU_TLBIVAH	WO	*SMMU_TLBIVAH, Invalidate Hyp TLB by VA on page 10-150*	-
0x0007C	Reserved	-	-	-
0x00080-0x0009C	IMPLEMENTATION DEFINED	-	Reserved for TLB Debug features	-
0x000A0-0x000FC	Reserved	-	-	-
0x00100	SMMU_sGATS1UR	WO	*SMMU_sGATS1UR, GAT Stage 1 Unprivileged Read on page 10-126*	-
0x00104	Reserved	-	-	-
0x00108	SMMU_sGATS1UW	WO	*SMMU_sGATS1UW, GAT Stage 1 Unprivileged Write on page 10-127*	-
0x0010C	Reserved	-	-	-
0x00110	SMMU_sGATS1PR	WO	*SMMU_sGATS1PR, GAT Stage 1 Privileged Read on page 10-125*	-
0x00114	Reserved	-	-	-
0x00118	SMMU_sGATS1PW	WO	*SMMU_sGATS1PW, GAT Stage 1 Privileged Write on page 10-126*	-
0x0011C	Reserved	-	-	-
0x00120	SMMU_sGATS12UR	WO	*SMMU_sGATS12UR, GAT Stages 1 and 2 Unprivileged Read on page 10-128*	-
0x00124	Reserved	-	-	-
0x00128	SMMU_sGATS12UW	WO	*SMMU_sGATS12UW, GAT Stages 1 and 2 Unprivileged Write on page 10-129*	-
0x0012C	Reserved	-	-	-
0x00130	SMMU_sGATS12PR	WO	*SMMU_sGATS12PR, GAT Stages 1 and 2 Privileged Read on page 10-127*	-
0x00134	Reserved	-	-	-
0x00138	SMMU_sGATS12PW	WO	*SMMU_sGATS12PW, GAT Stages 1 and 2 Privileged Write on page 10-128*	-
0x0013C	Reserved	-	-	-

Table 10-1 System MMU Global Register Space 0 (continued)

Offset	Name	Type	Description	Notes
0x00140-0x0017C	Reserved	-	-	-
0x00180	SMMU_sGPAR[31:0]	RW	*SMMU_sGPAR, Global Physical Address Register on page 10-135*	Banked with security
0x00184	SMMU_sGPAR[63:32]			
0x00188	SMMU_sGATSR	RO	*SMMU_sGATSR, Global Address Translation Status Register on page 10-130*	Banked with security
0x0018C-0x003FC	Reserved	-	-	-
0x00400	SMMU_NSCR0	RW	Secure alias for Non-secure copy of *SMMU_sCR0, Configuration Register 0 on page 10-120*	Secure only
0x00404	Reserved	-	-	-
0x00408	SMMU_NSCR2	RW	Secure alias for Non-secure copy of *SMMU_sCR2, Configuration Register 2 on page 10-125*	Secure only
0x0040C	Reserved	-	-	-
0x00410	SMMU_NSACR	RW	Secure alias for Non-secure copy of *SMMU_sACR, Auxiliary Configuration Register on page 10-120*	Secure only
0x00414-0x0041C	Reserved	-	-	-
0x00420-0x0043C	Reserved	-	-	-
0x00440	SMMU_NSGFAR[31:0]	RW	Secure alias for Non-secure copy of *SMMU_sGFAR, Global Fault Address Register on page 10-130*	Secure only, 64-bit
0x00444	SMMU_NSGFAR[63:32]			
0x00448	SMMU_NSGFSR	RW	Secure alias for Non-secure copy of *SMMU_sGFSR, Global Fault Status Register on page 10-131*	Secure only
0x0044C	SMMU_NSGFSRRESTORE	WO	Secure alias for Non-secure copy of *SMMU_sGFSRRESTORE, Global Fault Status Restore Register on page 10-132*	Secure only
0x00450	SMMU_NSGFSYNR0	RW	Secure alias for Non-secure copy of *SMMU_sGFSYNR0, Global Fault Syndrome Register 0 on page 10-133*	Secure only
0x00454	SMMU_NSGFSYNR1	RW	Secure alias for Non-secure copy of *SMMU_sGFSYNR1, Global Fault Syndrome Register 1 on page 10-134*	Secure only
0x00458	SMMU_NSGFSYNR2	RW	Secure alias for Non-secure copy of *SMMU_sGFSYNR2, Global Fault Syndrome Register 2 on page 10-135*	Secure only
0x0045C - 0x0046C	Reserved	-	-	-
0x00470	SMMU_NSTLBGSYNC	WO	Secure alias for Non-secure copy of *SMMU_sTLBGSYNC, Global Synchronize TLB Invalidate on page 10-139*	Secure only

Table 10-1 System MMU Global Register Space 0 (continued)

Offset	Name	Type	Description	Notes
0x00474	SMMU_NSTLBGSTATUS	RO	Secure alias for Non-secure copy of *SMMU_sTLBGSTATUS, Global TLB Status register on page 10-138*	Secure only
0x00478-0x0047C	Reserved	-	-	-
0x00480-0x0049C	IMPLEMENTATION DEFINED	-	Reserved for Secure alias to Non-secure TLB debug features	Secure only
0x004A0-0x004FC	Reserved	-	-	-
0x00500	SMMU_NSGATS1UR	WO	Secure alias for Non-secure copy of *SMMU_sGATS1UR, GAT Stage 1 Unprivileged Read on page 10-126*	Secure only
0x00504	Reserved	-	-	-
0x00508	SMMU_NSGATS1UW	WO	Secure alias for Non-secure copy of *SMMU_sGATS1UW, GAT Stage 1 Unprivileged Write on page 10-127*	Secure only
0x0050C	Reserved	-	-	-
0x00510	SMMU_NSGATS1PR	-	Secure alias for Non-secure copy of *SMMU_sGATS1PR, GAT Stage 1 Privileged Read on page 10-125*	Secure only
0x00514	Reserved	-	-	-
0x00518	SMMU_NSGATS1PW	WO	Secure alias for Non-secure copy of *SMMU_sGATS1PW, GAT Stage 1 Privileged Write on page 10-126*	Secure only
0x0051C	Reserved	-	-	-
0x00520	SMMU_NSGATS12UR	WO	Secure alias for Non-secure copy of *SMMU_sGATS12UR, GAT Stages 1 and 2 Unprivileged Read on page 10-128*	Secure only
0x00524	Reserved	-	-	-
0x00528	SMMU_NSGATS12UW	WO	Secure alias for Non-secure copy of *SMMU_sGATS12UW, GAT Stages 1 and 2 Unprivileged Write on page 10-129*	Secure only
0x0052C	Reserved	-	-	-
0x00530	SMMU_NSGATS12PR	WO	Secure alias for Non-secure copy of *SMMU_sGATS12PR, GAT Stages 1 and 2 Privileged Read on page 10-127*	Secure only
0x00534	Reserved	-	-	-
0x00538	SMMU_NSGATS12PW	WO	Secure alias for Non-secure copy of *SMMU_sGATS12PW, GAT Stages 1 and 2 Privileged Write on page 10-128*	Secure only

Table 10-1 System MMU Global Register Space 0 (continued)

Offset	Name	Type	Description	Notes
0x0053C-0x0057C	Reserved	-	-	-
0x0580	SMMU_NSGPAR[31:0]	RW	Secure alias for Non-secure copy of *SMMU_sGPAR, Global Physical Address Register on page 10-135*	Secure only
0x00584	SMMU_NSGPAR[63:32]			
0x00588	SMMU_NSGATSR	RO	Secure alias for Non-secure copy of *SMMU_sGATSR, Global Address Translation Status Register on page 10-130*	Secure only
0x0058C-0x007FC	Reserved	-	-	-
0x00800	SMMU_SMR0	RW	*SMMU_SMRn, Stream Match Register on page 10-149*	RAZ/WI in an implementation with StreamID indexing
0x00804	SMMU_SMR1			
0x00808-0x009FC	SMMU_SMR2 to SMMU_SMR127			
0x00A00-0x00BFC	Reserved	-	-	-
0x00C00	SMMU_S2CR0	RW	*SMMU_S2CRn, Stream-to-Context Register on page 10-139*	-
0x00C04	SMMU_S2CR1			
0x00C08-0x00DFC	SMMU_S2CR2 to SMMU_S2CR127			
0x00E00 - (*PAGESIZE* −0x4)	Reserved	-	-	-

10.2 Reset values

Table 10-2 shows the register field values in System MMU Global Register Space 0 following a system reset. The fields that do not appear in Table 10-2 reset to UNKNOWN values.

Table 10-2 Reset values

Field	Reset	Notes
SMMU_sCR0.CLIENTPD	0b1	Client transaction bypasses System MMU translation.
SMMU_sCR0.GFIE	0b0	For SMMU_CR0.GFIE, disable assertion of interrupt from UNPREDICTABLE SMMU_GFSR content. For SMMU_SCR0.GFIE, disable assertion of interrupt from UNPREDICTABLE SMMU_SGFSR content. See *SMMU_sGFSR, Global Fault Status Register on page 10-131*.
SMMU_sCR0.GFRE	0b0	For SMMU_CR0.GFRE, disable spurious abort from UNPREDICTABLE SMMU_GFSR content. For SMMU_SCR0.GFRE, disable spurious abort from UNPREDICTABLE SMMU_SGFSR content. See *SMMU_sGFSR, Global Fault Status Register on page 10-131*.
SMMU_sCR0.GCFGFIE	0b0	For SMMU_CR0.GCFGFIE, disable assertion of interrupt from UNPREDICTABLE SMMU_GFSR content. For SMMU_SCR0.GCFGFIE, disable assertion of interrupt from UNPREDICTABLE SMMU_SGFSR content. See *SMMU_sGFSR, Global Fault Status Register on page 10-131*.
SMMU_sCR0.GCFGFRE	0b0	For SMMU_CR0.GCFGFRE, disable spurious abort from UNPREDICTABLE SMMU_GFSR content. For SMMU_SCR0.GCFGFRE, disable spurious abort from UNPREDICTABLE SMMU_SGFSR content. See *SMMU_sGFSR, Global Fault Status Register on page 10-131*.
SMMU_sCR0.GSE	0b0	Disable global stalling across Translation context banks.
SMMU_sCR0.MTCFG	0b0	No override of input memory attributes.
SMMU_sCR0.SHCFG	0b00	No override of input shareability attributes.
SMMU_sCR0.RACFG	0b00	No override of input read allocate attribute.
SMMU_sCR0.WACFG	0b00	No override of input write allocate attribute.
SMMU_sCR0.TRANSIENTCFG	0b00	No override of input transient allocate attribute.
SMMU_SCR0.STALLD	0b0	Permit per context stalling on context faults. See *SMMU_sCR0, Configuration Register 0 on page 10-120*.
SMMU_SCR0.NSCFG	0b00	No override of input NS-Attr. See *SMMU_sCR0, Configuration Register 0 on page 10-120*.
SMMU_sTLBGSTATUS.GSACTIVE	0b0	No active Global TLB Invalidate operation Sync.
SMMU_SCR1.NSNUMCBO	Implemented NUMCB	Do not require Secure software to write NSNUMCBO if such software is not using any System MMU context.
SMMU_SCR1.GASRAE	0b0	Permit Secure and Non-secure configuration accesses.
SMMU_SCR1.SPMEN	0b0	No contribution to performance monitor counters from any Secure transaction processing event.

Table 10-2 Reset values (continued)

Field	Reset	Notes
SMMU_SCR1.SIF	0b0	Secure state instruction fetch permitted to Non-secure memory.
SMMU_SCR1.NSNUMIRPTO	Implemented NUMIRPT	It is IMPLEMENTATION DEFINED whether this field is read-only. See *SMMU_SCR1, Secure Configuration Register 1 on page 10-145*.
SMMU_SCR1.NSNUMSMRGO	Implemented NUMSMRG	-

10.3 Secure alias for Non-secure registers

In an implementation that includes the Security Extensions, additional alias registers and commands are provided at different offset addresses, permitting Secure software to access Non-secure versions of the resources.

In general, the aliases are accessed in the following usage cases:

Secure software manages Non-secure context resources

When the SMMU_SCR1.GASRAE bit is set to 1, the Global address space is only accessible by Secure configuration memory accesses, This means that Non-secure software must request that Secure software performs certain operations. In such cases it might be necessary for Secure software to access any of:

- SMMU_NSCR0
- SMMU_NSCR2
- SMMU_NSACR
- SMMU_NSGFAR
- SMMU_NSGFSR
- SMMU_NSGFSRRESTORE
- SMMU_NSGFSYNRn
- SMMU_NSGPAR
- SMMU_NSGATSR
- SMMU_NSGATS1*.

For example, a Secure access to SMMU_NSGFSYNR1 is the only mechanism for obtaining the SSD_Index for a faulty Non-secure transaction.

The SMMU architecture provides separate Secure and Non-secure address translation resources. If Secure software is managing the Non-secure resources, it can use the Non-secure address translation commands, and the Secure resources remain available for Secure software to use as required..

Note

Although the SMMU architecture provides separate Secure and Non-secure TLB Invalidation resources, Secure software cannot initiate a TLB Invalidate operation directly using Non-secure resources. In this usage case, Secure software must use Secure resources to initiate these operations.

Save and restore, or Powerdown operations

During save and restore operations, such as before Powerdown, Secure software can save the Non-secure state by accessing the following aliases:

- SMMU_NSCR0
- SMMU_NSCR2
- SMMU_NSACR
- SMMU_NSGFAR
- SMMU_NSGFSR
- SMMU_NSGFSRRESTORE
- SMMU_NSGFSYNRn
- SMMU_NSGPAR.

During Powerdown, Secure software must ensure that the SMMU is quiescent by:

- completing all TLB Invalidate operations
- completing all address translation operations
- ensuring that all upstream devices are quiescent.

To ensure that any queued Non-secure TLB Invalidate operations are issued and completed, Secure software accesses SMMU_NSTLBGSYNC to start the operations, then polls SMMU_NSTLBGSTATUS to check for completion. Similarly, Secure software polls SMMU_NSGATSR to ensure that all Non-secure address translation operations are complete.

Note

When Secure software issues a SMMU_TLBIALLNSNH or SMMU_TLBIALLH operation, it uses Secure TLB Invalidate resources, and manages them using SMMU_STLBGSYNC and SMMU_STLBGSTATUS. This is distinct from a requirement to use SMMU_NSTLBGSYNC or SMMU_NSTLBGSTATUS. In general, the Secure alias must only be used when an operation is not otherwise possible.

10.4 Memory attribute, MemAttr

The MemAttr field is common to a number of registers. MemAttr[3:2] gives a top-level definition of the memory type, and of the cacheability of a Normal memory region. See Table 10-3.

The encoding of MemAttr[1:0] depends on the memory type indicated by MemAttr[3:2]:

- when MemAttr[3:2] == `0b00`, indicating Strongly-ordered or Device memory, Table 10-4 shows the encoding of MemAttr[1:0]
- when MemAttr[3:2] != `0b00`, indicating Normal memory, Table 10-5 shows the encoding of MemAttr[1:0].

Table 10-3 MemAttr[3:2] encoding

MemAttr[3:2]	Memory type	Cacheability
`0b00`	Strongly-ordered or Device, determined by MemAttr[1:0]	Not applicable
`0b01`	Normal	Outer Non-cacheable
`0b10`		Outer Write-Through Cacheable
`0b11`		Outer Write-Back Cacheable

Table 10-4 MemAttr[1:0] encoding for Strongly-ordered or Device memory

MemAttr[1:0]	Meaning when MemAttr[3:2] == `0b00`
`0b00`	Strongly-ordered memory
`0b01`	Device memory
`0b10`	UNPREDICTABLE
`0b11`	UNPREDICTABLE

Table 10-5 MemAttr[1:0] encoding for Normal memory

MemAttr[1:0]	Meaning when MemAttr[3:2] != `0b00`
`0b00`	UNPREDICTABLE
`0b01`	Inner Non-cacheable
`0b10`	Inner Write-Through Cacheable
`0b11`	Inner Write-Back Cacheable

10.5 Multi-format registers and reserved fields

Some registers have a number of possible formats, where the format depends on:

- the format that the register has been configured to
- the format of the data recorded in the register.

Several register formats have a TYPE field, that has different encodings for different registers, and some registers are written with data whose format depends on the outcome of the operation for which the result is to be written.

For such a register, where a field is reserved:

- A read must return the value that was last successfully written to that field, regardless of how the register is used. If the field has not been written to since reset, the read must return the specified reset value, if one exists, or an UNKNOWN value. See *Reset values on page 10-109*.
- A write must update a storage location associated with the written field.
- The state of the storage location associated with a field must have no other effect on the processor behavior, other than determining the value to be read back while the use of the register means that the field is reserved.

If the TYPE field changes so that a reserved field becomes a field with defined behavior, the value last written to that field takes effect as specified by the individual register descriptions.

Each behavior only applies to a field that is:

- reserved in one format with an allocated meaning in a different format
- being processed by hardware.

If a field is reserved regardless of how the register is used, the behavior is unchanged from the behavior described in the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition*.

A field processed by software is to be preserved if appropriate, or written to with the value required by the *should be* value if being written with a new value. Otherwise, the effect is UNPREDICTABLE.

10.6 System MMU Global Register Space 0 register descriptions

This section describes all of the Global Register Space 0 registers that might be present in a System MMU implementation. Registers are shown in register name order.

The names of some registers indicates the security of the register. For more information, see:

- *Register names on page xiii*
- *Banked registers on page 2-22*.

10.6.1 SMMU_IDR0-7, Identification registers

The SMMU_IDR0-7 characteristics are:

Purpose Provides System MMU capability information.

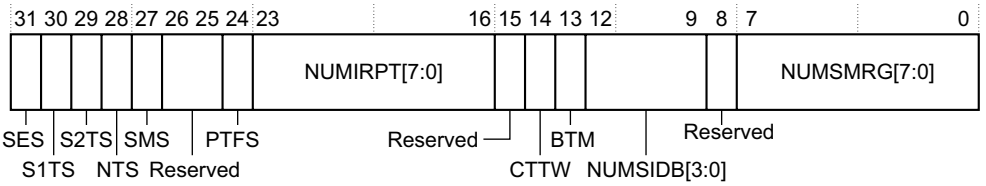
Usage constraints There are no usage constraints.

Configurations In an implementation that includes the Security Extensions, the behavior of some fields depends on whether Secure or Non-secure software is accessing the register. See the field descriptions for more information.

Attributes 32-bit RO registers with UNKNOWN reset values.

SMMU_IDR0

The SMMU_IDR0 bit assignments are:



SES, bit[31] Security Extensions Support. The possible values of this bit are:

0 The implementation does not include the Security Extensions.

1 The implementation includes the Security Extensions.

This field is RAZ for Non-secure reads of SMMU_IDR0.

S1TS, bit[30] Stage 1 Translation Support. The possible values of this bit are:

0 Stage 1 translation is unsupported.

1 Stage 1 translation is supported.

If the NTS bit is set to 1, this bit must also be set to 1.

S2TS, bit[29] Stage 2 Translation Support. The possible values of this bit are:

0 Stage 2 translation is unsupported.

1 Stage 2 translation is supported.

This field only applies to Non-secure client transactions.

If the NTS bit is set to 1, this bit must also be set to 1.

NTS, bit[28] Nested Translation Support. The possible values of this bit are:

0 Nested translation is not supported.

1 Nested translation is supported.

This field only applies to Non-secure client transactions.

Note

If this bit is set to 1, then both the S1TS and S2TS bits must be set to 1 also.

SMS, bit[27] Stream Match Support. The possible values of this bit are:

0 Stream Match Register functionality is not included.

1 Stream Match Register functionality is included.

Bits[26:25] Reserved.

PTFS, bit[24] Support for translation table formats. The possible values of this bit are:

0 Short-descriptor and Long-descriptor formats are supported.

1 Long-descriptor format is supported.

NUMIRPT[7:0], bits[23:16]

Number of implemented Context fault interrupts.

Indicates the number of Context fault interrupts supported by the System MMU, in the range 0-128. NUMIRPT==0 is permitted in an implementation that does not provide a Translation context bank.

In an implementation that includes the Security Extensions, access to this field by Non-secure software gives the value configured in SMMU_SCR1.NSNUMIRPTO.

Bit[15] Reserved.

CTTW, bit[14] Coherent Translation Table Walk. The possible values of this bit are:

0 Coherent translation table walks are unsupported.

1 Coherent translation table walks are supported.

BTM, bit[13] Broadcast TLB Maintenance. The possible values of this bit are:

0 Broadcast TLB maintenance is unsupported.

1 Broadcast TLB maintenance is supported.

NUMSIDB[3:0], bits[12:9]

Number of StreamID Bits.

Indicates the number of implemented StreamID bits, in the range 0-15.

Bit[8] Reserved.

NUMSMRG[7:0], bits[7:0]

Number of Stream Mapping Register Groups.

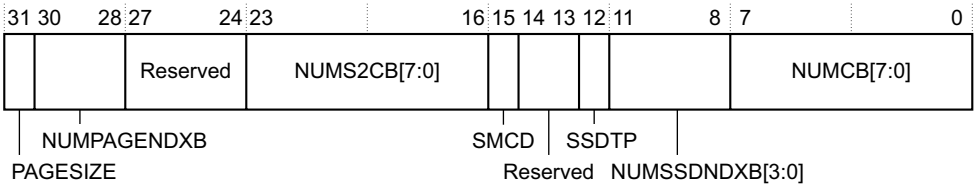
Indicates the number of Stream mapping register groups in the Stream match table, in the range 0-127.

In an implementation that includes Stream matching, the value of this field is greater than or equal to 1.

In an implementation that includes the Security Extensions, access to this field by Non-secure software gives the value configured in SMMU_SCR1.NSNUMSMRGO.

SMMU_IDR1

The SMMU_IDR1 bit assignments are:



PAGESIZE, bit[31]

System MMU Page Size.

Indicates the size of each page in the System MMU register map.

The possible values of this bit are:

0 4KB.

1 64KB.

NUMPAGENDXB, bits[30:28]

System MMU Number of Page Index Bits.

Indicates how many *PAGESIZE* pages occupy the global address space or the translation context address space, where `NUMPAGE = 2(SMMU_IDR.NUMPAGENDXB + 1)`.

Bits[27:24] Reserved.

NUMS2CB[7:0], bits[23:16]

Number of Stage 2 Context Banks.

Indicates the number of Translation context banks that only support the stage 2 translation format, in the range 0-127.

This field is validated by SMMU_IDR0.S2TS.

SMCD, bit[15] Stream Match Conflict Detection.

The possible values of this bit are:

0 The detection of all Stream match conflicts is not guaranteed.

1 The detection of all Stream match conflicts is guaranteed.

See *StreamID matching on page 2-25* for more information about stream matching.

Bits[14:13] Reserved.

SSDTP, bit[12] Security State Determination Table Present. The possible values of this bit are:

0 The Security state determination address space is UNK/WI.

1 The Security state determination address space is populated.

In an implementation that includes the Security Extensions, Non-secure access to this field is RAZ.

NUMSSDNDXB[3:0], bits[11:8]

Number of SSD_Index bits.

Indicates the number of SSD_Index bits for indexing the security state determination table. This field is only valid if SSDTP==1. Otherwise, it is reserved.

In an implementation that includes the Security Extensions, Non-secure access to this field is RAZ.

NUMCB, bits[7:0]

Number of Context Banks.

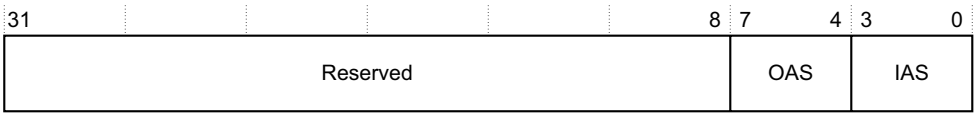
Indicates the total number of implemented Translation context banks, in the range 0-128.

The value reported in NUMCB includes Translation context banks that only support the stage 2 format. If an implementation includes stage 1 translation, the number of Translation context banks that support the stage 1 format is given by SMMU_IDR1.NUMCB – SMMU_IDR1.NUMS2CB.

In an implementation that includes the Security Extensions, a read of this field by Non-secure software gives the value configured in SMMU_SCR1.NSNUMCBO.

SMMU_IDR2

The SMMU_IDR2 bit assignments are:



Bits[31:8] Reserved.

OAS, bits[7:4] Output Address Size. The encoding of this field is:

`0b0000` 32-bit output address size.

`0b0001` 36-bit output address size.

`0b0010` 40-bit output address size.

All other encodings are reserved.

IAS, bits[3:0] Input Address Size. The encoding of this field is:

`0b0000` 32-bit input address size.

`0b0001` 36-bit input address size.

`0b0010` 40-bit input address size.

All other encodings are reserved.

SMMU_IDR3

The SMMU_IDR3 bit assignments are reserved.

SMMU_sIDR4-5

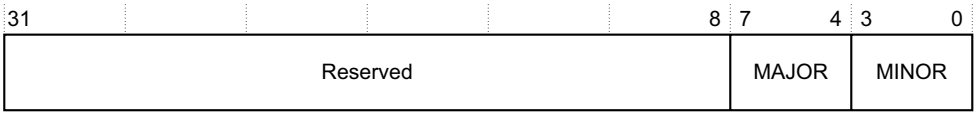
The SMMU_IDR4-5 bit assignments are IMPLEMENTATION DEFINED.

SMMU_IDR6

The SMMU_IDR6 bit assignments are reserved.

SMMU_IDR7

The SMMU_IDR7 bit assignments are:



Bits[31:8] Reserved.

MAJOR, bits[7:4] The major part of the implementation version number.

MINOR, bits[3:0] The minor part of the implementation version number.

10.6.2 SMMU_sACR, Auxiliary Configuration Register

The SMMU_*s*ACR characteristics are:

Purpose Provides IMPLEMENTATION DEFINED functionality.

Usage constraints There are no usage constraints.

Configurations In an implementation that includes the Security extensions, this register is Banked.

SMMU_NSACR is provided as a Secure alias of the Non-secure SMMU_ACR. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The SMMU_*s*ACR bit assignments are IMPLEMENTATION DEFINED.

10.6.3 SMMU_sCR0, Configuration Register 0

The SMMU_*s*CR0 characteristics are:

Purpose Provides top-level control of the System MMU.

Usage constraints The *Non-secure* register, SMMU_CR0, does not provide full top-level control of the System MMU for Secure transactions.

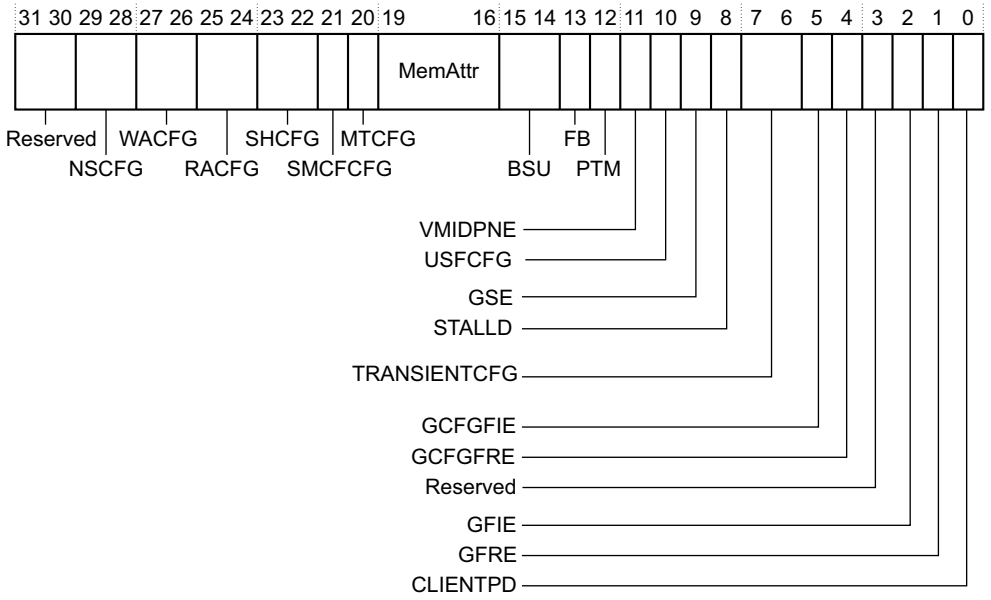
Configurations In an implementation that includes the Security Extensions:

- some SMMU_CR0 fields only apply to Non-secure transactions
- this register is Banked.

SMMU_NSCR0 is provided as a Secure alias of the Non-secure SMMU_CR0. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit RW register. See the field descriptions for information about the reset values.

The SMMU_*s*CR0 bit assignments are:



Bits[31:30] Reserved.

NSCFG, bits[29:28]

Non-secure Configuration.

This field only exists in SMMU_SCR0. In SMMU_CR0, these bits are reserved.

This field only applies to Secure transactions bypassing the System MMU stream mapping process. See *Bypassing the Stream mapping table on page 2-26*.

The encoding of this field is:

`0b00` Use the default NS attribute

`0b01` Reserved

`0b10` Secure

`0b11` Non-secure.

These bits reset to 0.

WACFG, bits[27:26]

Write-Allocate Configuration, controls the allocation hint for write accesses.

This field applies to transactions that bypass the Stream mapping table. See *Bypassing the Stream mapping table on page 2-26*.

The encoding of this field is:

`0b00` Default attributes.

`0b01` Reserved.

`0b10` Write-Allocate.

`0b11` No Write-Allocate.

These bits reset to 0.

RACFG, bits[25:24]

Read-Allocate Configuration. Controls the allocation hint for read accesses.

Applies to transactions that bypass the Stream mapping table. See *Bypassing the Stream mapping table on page 2-26*.

The encoding of this field is:

`0b00` Default allocation attributes.

`0b01` Reserved.

`0b10` Read-Allocate.

`0b11` No Read-Allocate.

These bits reset to 0.

SHCFG, bits[23:22] Shared Configuration.

Applies to transactions that bypass the Stream mapping table. See *Bypassing the Stream mapping table on page 2-26*.

The encoding of this field is:

`0b00` Default Shareable attribute.

`0b01` Outer Shareable.

`0b10` Inner Shareable.

`0b11` Non-shareable.

These bits reset to 0.

SMCFCFG, bit[21]

Stream Match Conflict Fault Configuration. Controls transactions with multiple matches in the Stream mapping table.

The possible values of this bit are:

`0` Permit the transaction to bypass the System MMU.

`1` Raise a Stream match conflict fault.

It is IMPLEMENTATION DEFINED whether:

- The System MMU guarantees the detection of every Stream match conflict. See *StreamID matching on page 2-25* for more information.
- Stream match conflict handling is configurable. If not configurable, the value of SMCFCFG is fixed and writes are ignored.

This bit resets to an UNKNOWN value.

MTCFG, bit[20]

Memory Type Configuration, applies to transactions that bypass the Stream mapping table. See *Bypassing the Stream mapping table on page 2-26*.

The possible values of this bit are:

`0` Use the default memory attributes.

`1` Use the MemAttr field for memory attributes.

This bit resets to 0.

MemAttr, bits[19:16]

Memory Attributes. See *Memory attribute, MemAttr on page 10-113*.

If SMMU_CR0.MTCFG==1, the memory attributes can be overlaid.

These bits reset to an UNKNOWN value.

BSU, bits[15:14]

Barrier Shareability Upgrade.

Upgrades the required shareability domain of barriers issued by client devices that are not mapped to a Translation context bank, by setting the minimum shareability domain applied to any barrier.

Applies to transactions bypassing the Stream mapping table. See *Bypassing the Stream mapping table on page 2-26*.

The encoding of this field is:

`0b00` No effect.

`0b01` Inner Shareable.

`0b10` Outer Shareable.

`0b11` Full system.

This functionality might not be supported by all system topologies. In an implementation that does not support it, BSU is RAZ/SBZP.

These bits reset to an UNKNOWN value.

FB, bit[13] Force Broadcast of TLB, branch predictor and instruction cache maintenance operations. Applies to transactions bypassing the Stream mapping table. See *Bypassing the Stream mapping table on page 2-26*.

Affects client TLB maintenance, BPIALL and ICIALLU operations. If FB==1, any affected operation is 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.

This bit resets to an UNKNOWN value.

PTM, bit[12] Private TLB Maintenance. The possible values of this bit are:

0 The System MMU participates in broadcast TLB maintenance with the wider system, if supported in the implementation and as indicated by SMMU_IDR0.BTM.

1 System MMU TLBs are privately managed and are not required to respond to broadcast TLB maintenance operations from the wider system.

The PTM field is a hint. A broadcast TLB Invalidate operation is still permitted to affect all cached translations that are unlocked in the System MMU.

This bit resets to an UNKNOWN value.

VMIDPNE, bit[11]

VMID Private Namespace Enable. The possible values of this bit are:

0 System MMU values are coordinated with the wider system.

1 System MMU VMID values are a private namespace, not coordinated with the wider system.

If VMIDPNE==1, broadcast TLB Invalidate operations specifying a VMID value are not required to apply to cached translations in the system MMU.

The VMIDPNE field is a hint. A broadcast TLB Invalidate operation is still permitted to affect all cached translations that are unlocked in the System MMU.

This bit resets to an UNKNOWN value.

In SMMU_SCR0, VMIDPNE is reserved.

USFCFG, bit[10]

Unidentified Stream Fault Configuration. The possible values of this bit are:

0 Permit any transaction that does not match any entries in the Stream mapping table to pass through.

1 Raise an Unidentified stream fault on any transaction that does not match any Stream mapping table entries.

This bit resets to an UNKNOWN value.

GSE, bit[9] 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 field is only writable if the implementation has global stall behavior. Otherwise, it is RAZ/WI.

This bit resets to 0.

STALLD, bit[8]

Stall Disable. The possible values of this bit are:

0 Permit per-context stalling on context faults.

1 Disable per-context stalling on context faults.

Setting this bit to 1 disables SMMU_sCR0.GSE and causes SMMU_sCR0.GSE to be RAZ/WI.

In an implementation that includes the Security Extensions, SMMU_CR0.STALLD must apply to a Non-secure Translation context bank, and must affect SMMU_CR0.GSE. It is permitted to apply to a Secure Translation context bank, and is permitted to affect SMMU_SCR0.GSE.

In an implementation that includes the Security Extensions, this bit resets to 0. In an implementation that does not include the Security Extensions, it resets to an UNKNOWN value.

TRANSIENTCFG, bits[7:6]

Transient Configuration, controls the transient allocation hint.

Applies to any transaction that bypasses the Stream mapping table. See *Bypassing the Stream mapping table on page 2-26*.

The encoding of this field is:

`0b00` Default transient allocation attributes.

`0b01` Reserved.

`0b10` Non-transient.

`0b11` Transient.

These bits reset to 0.

It is IMPLEMENTATION DEFINED whether this field is present. If not present, these bits are RAZ/WI.

GCFGFIE, bit[5]

Global Configuration Fault Interrupt Enable. The possible values of this bit are:

0 Do not raise an interrupt on a Global configuration fault.

1 Raise an interrupt on a Global configuration fault.

This bit resets to 0.

GCFGFRE, bit[4]

Global Configuration Fault Report Enable. The possible values of this bit are:

0 Do not return an abort on a Global configuration fault.

1 Return an abort on a Global configuration fault.

This bit resets to 0.

Bit[3] Reserved.

GFIE, bit[2] Global Fault Interrupt Enable. The possible values of this bit are:

0 Do not raise an interrupt on a Global fault.

1 Raise an interrupt on a Global fault.

This bit resets to 0.

GFRE, bit[1] Global Fault Report Enable. The possible values of this bit are:

0 Do not return an abort on a Global fault.

1 Return an abort on a Global fault.

This bit resets to 0.

For exclusive access transactions, when this bit is set to 0, ARM recommends that the SMMU reports the transaction as failed. See *Reporting exclusive access transactions on page 3-61* for more information.

CLIENTPD, bit[0]

Client Port Disable. The possible values of this bit are:

0 Each System MMU client access is subject to System MMU translation, access control, and attribute generation.

1 Each System MMU client access bypasses System MMU translation, access control, and attribute generation.

This bit resets to 1.

10.6.4 SMMU_*s*CR2, Configuration Register 2

The SMMU_*s*CR2 characteristics are:

Purpose Provides top-level control of the System MMU.

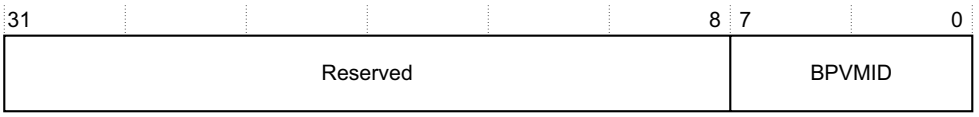
Usage constraints There are no usage constraints.

Configurations In an implementation that includes the Security Extensions, this register is Banked.

SMMU_NSCR2 is provided as a Secure alias of the Non-secure SMMU_CR2. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit RW register.

The SMMU_*s*CR2 bit assignments are:



Bits[31:8] Reserved.

BPVMID, bits[7:0]

Bypass VMID.

This field is reserved for IMPLEMENTATION DEFINED use as a VMID field applied to client transactions that bypass the System MMU. See *Bypassing the Stream mapping table on page 2-26*.

Whether this field is implemented, and the effect it has, is IMPLEMENTATION DEFINED.

In an implementation that does not support this field, these bits are RAZ/WI.

If an implementation that includes the Security Extensions does not provide a VMID for Secure translation regimes, any Secure VMID field is ignored. See *Interaction with the Security Extensions on page 2-32* for more information.

10.6.5 SMMU_*s*GATS1PR, GAT Stage 1 Privileged Read

The Global Address Translation Stage 1 Privileged Read characteristics are:

Purpose Translates an argument-supplied address. Writes the result to SMMU_sGPAR.

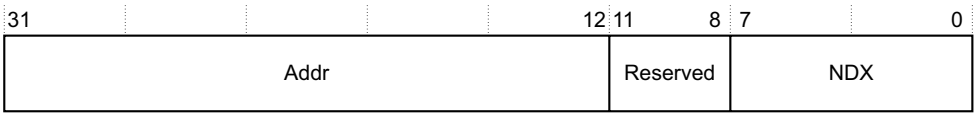
Usage constraints The security state of the transaction determines whether SMMU_GATS1PR or SMMU_SGATS1PR is used.

Configurations If the implementation includes the Security Extensions, this command is in SMMU_SGATS1PR. Otherwise, it is in SMMU_GATS1PR.

SMMU_NSGATS1PR is provided as a Secure alias of the Non-secure SMMU_GATS1PR. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit WO command.

The SMMU_*s*GATS1PR bit assignments are:



Addr, bits[31:12]

Address translation operation.

The translation is performed as though the address is associated with a privileged read.

Bits[11:8] Reserved.

NDX, bits[7:0] Translation context bank index for stage 1 translation.

See *Address translation commands in the global address space on page 4-68*.

10.6.6 SMMU_*s*GATS1PW, GAT Stage 1 Privileged Write

The Global Address Translation Stage 1 Privileged Write characteristics are:

Purpose Translates an argument-supplied input address. Writes the result to SMMU_sGPAR.

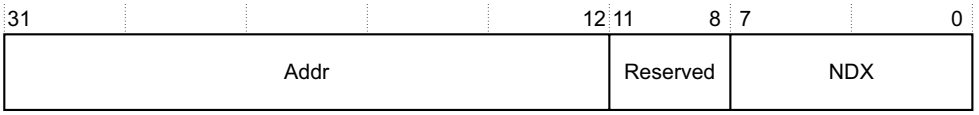
Usage constraints The security state of the transaction determines whether SMMU_GATS1PW or SMMU_SGATS1PW is used.

Configurations If the implementation includes the Security Extensions, this command is in SMMU_SGATS1PW. Otherwise, it is in SMMU_GATS1PW.

SMMU_NSGATS1PW is provided as a Secure alias of the Non-secure SMMU_GATS1PW. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit WO command.

The SMMU_*s*GATS1PW bit assignments are:



Addr, bits[31:12]

Address translation operation.

The translation is performed as though the address is associated with a privileged write.

Bits[11:8] Reserved.

NDX, bits[7:0] Translation context bank index for stage 1 translation.

See *Address translation commands in the global address space on page 4-68*.

10.6.7 SMMU_*s*GATS1UR, GAT Stage 1 Unprivileged Read

The Global Address Translation Stage 1 Unprivileged Read characteristics are:

Purpose Translates an argument-supplied input address. Writes the result to SMMU_sGPAR.

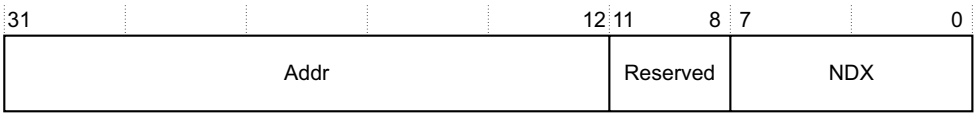
Usage constraints The security state of the transaction determines whether SMMU_GATS1UR or SMMU_SGATS1UR is used.

Configurations If the implementation includes the Security Extensions, this command is available in SMMU_SGATS1UR. Otherwise, it is in SMMU_GATS1UR.

SMMU_NSGATS1UR is provided as a Secure alias of the Non-secure SMMU_GATS1UR. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit WO command.

The SMMU_*s*GATS1UR bit assignments are:



Addr, bits[31:12] Address translation operation.

Performed as though the address is associated with an unprivileged read.

Bits[11:8] Reserved.

NDX, bits[7:0] Translation context bank index for stage 1 translation.

See *Address translation commands in the global address space on page 4-68*.

10.6.8 SMMU_*s*GATS1UW, GAT Stage 1 Unprivileged Write

The Global Address Translation Stage 1 Unprivileged Write characteristics are:

Purpose Translates an argument-supplied input address. Writes the result to SMMU_sGPAR.

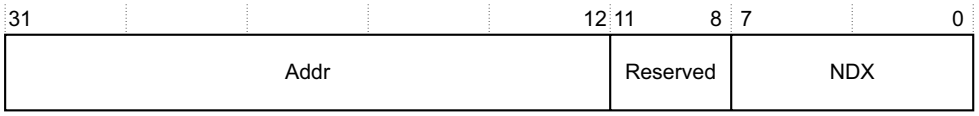
Usage constraints The security state of the transaction determines whether SMMU_GATS1UW or SMMU_SGATS1UW is used.

Configurations If the implementation includes the Security Extensions, this command is in SMMU_SGATS1UW. Otherwise, it is in SMMU_GATS1UW.

SMMU_NSGATS1UW is provided as a Secure alias of the Non-secure SMMU_GATS1UW. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit WO command.

The SMMU_*s*GATS1UW bit assignments are:



Addr, bits[31:12]

Address translation operation.

The translation is performed as though the address is associated with an unprivileged write.

Bits[11:8] Reserved.

NDX, bits[7:0] Translation context bank index for stage 1 translation.

See *Address translation commands in the global address space on page 4-68*.

10.6.9 SMMU_*s*GATS12PR, GAT Stages 1 and 2 Privileged Read

The Global Address Translation Stage 1 and Stage 2 Privileged Read characteristics are:

Purpose Translates an argument-supplied input address. Writes the result to SMMU_sGPAR.

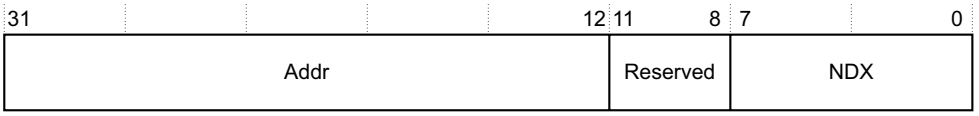
Usage constraints The security state of the transaction determines whether SMMU_GATS12PR or SMMU_SGATS12PR is used.

Configurations If the implementation includes the Security Extensions, this command is available in SMMU_SGATS12PR. Otherwise, it is in SMMU_GATS12PR.

SMMU_NSGATS12PR is provided as a Secure alias of the Non-secure SMMU_GATS12PR. See Table 10-1 on page 10-104. See also *Secure alias for Non-secure registers on page 10-111* for general information.

Attributes A 32-bit WO command.

The SMMU_*s*GATS12PR bit assignments are:



Addr, bits[31:12]

Address translation operation.

The translation is performed as though the address is associated with a privileged read.

Bits[11:8] Reserved.

NDX, bits[7:0] Translation context bank index for stage 1 translation.

See *Address translation commands in the global address space on page 4-68*.

10.6.10 SMMU_*s*GATS12PW, GAT Stages 1 and 2 Privileged Write

The Global Address Translation Stage 1 and Stage 2 Privileged Write characteristics are:

Purpose Translates an argument-supplied input address. Writes the result to SMMU_sGPAR.

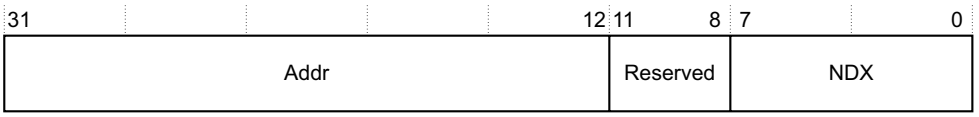
Usage constraints The security state of the transaction determines whether SMMU_GATS12PW or SMMU_SGATS12PW is used.

Configurations If the implementation includes the Security Extensions, this command is in SMMU_SGATS12PW. Otherwise, it is in SMMU_GATS12PW.

SMMU_NSGATS12PW is provided as a Secure alias of the Non-secure SMMU_GATS12PW. See Table 10-1 on page 10-104. See also *Secure alias for Non-secure registers on page 10-111* for general information.

Attributes A 32-bit WO command.

The SMMU_*s*GATS12PW bit assignments are:



Addr, bits[31:12]

Address translation operation.

The translation is performed as though the address is associated with a privileged write.

Bits[11:8] Reserved.

NDX, bits[7:0] Translation context bank index for stage 1 translation.

See *Address translation commands in the global address space on page 4-68*.

10.6.11 SMMU_*s*GATS12UR, GAT Stages 1 and 2 Unprivileged Read

The Global Address Translation Stage 1 and Stage 2 Unprivileged Read characteristics are:

Purpose Translates an argument-supplied input address. Writes the result to SMMU_sGPAR.

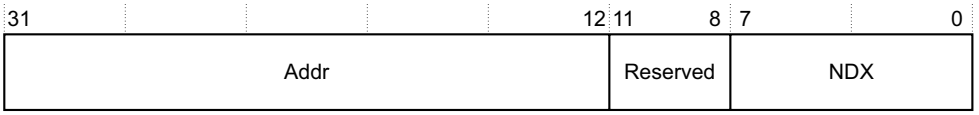
Usage constraints The security state of the transaction determines whether SMMU_GATS12UR or SMMU_SGATS12UR is used.

Configurations If the implementation includes the Security Extensions, this command is in SMMU_SGATS12UR. Otherwise, it is in SMMU_GATS12UR.

SMMU_NSGATS12UR is provided as a Secure alias of the Non-secure SMMU_GATS12UR. See Table 10-1 on page 10-104. See also *Secure alias for Non-secure registers on page 10-111* for general information.

Attributes A 32-bit WO command.

The SMMU_*s*GATS12UR bit assignments are:



Addr, bits[31:12]

Address translation operation.

The translation is performed as though the address is associated with an unprivileged read.

Bits[11:8] Reserved.

NDX, bits[7:0] Translation context bank index for stage 1 translation.

See *Address translation commands in the global address space on page 4-68*.

10.6.12 SMMU_*s*GATS12UW, GAT Stages 1 and 2 Unprivileged Write

The Global Address Translation Stage 1 and Stage 2 Unprivileged Write characteristics are:

Purpose Translates an argument-supplied input address. Writes the result to SMMU_sGPAR.

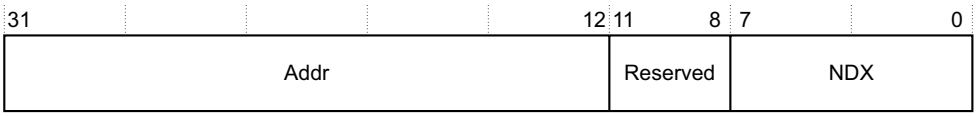
Usage constraints The security state of the transaction determines whether SMMU_GATS12UW or SMMU_SGATS12UW is used.

Configurations If the implementation includes the Security Extensions, this command is in SMMU_SGATS12UW. Otherwise, it is in SMMU_GATS12UW.

SMMU_NSGATS12UW is provided as a Secure alias of the Non-secure SMMU_GATS12UW. See Table 10-1 on page 10-104. See also *Secure alias for Non-secure registers on page 10-111* for general information.

Attributes A 32-bit WO command.

The SMMU_*s*GATS12UW bit assignments are:



Addr, bits[31:12]

Address translation operation.

The translation is performed as though the address is associated with an unprivileged write.

Bits[11:8] Reserved.

NDX, bits[7:0] Translation context bank index for stage 1 translation.

See *Address translation commands in the global address space on page 4-68*.

10.6.13 SMMU_*s*GATSR, Global Address Translation Status Register

The SMMU_*s*GATSR characteristics are:

Purpose Gives status information pertaining to active global address translation operations.

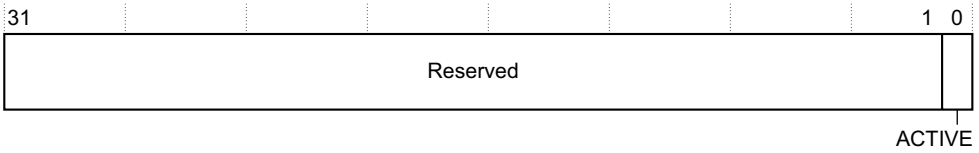
Usage constraints If software requests a new address translation operation while an existing operation is active, the result is UNPREDICTABLE. See *Usage model on page 4-68* for more information.

Configurations If the implementation includes the Security Extensions, this register is Banked.

SMMU_NSGATSR is provided as a Secure alias of the Non-secure SMMU_GATSR. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit RO register with an UNKNOWN reset value.

The SMMU_*s*GATSR bit assignments are:



Bits[31:1] Reserved.

ACTIVE, bit[0]

Address Translation Active.

The possible values of this bit are:

0 Address translation not active.

1 Address translation active, SMMU_sGPAR is yet to be updated.

10.6.14 SMMU_*s*GFAR, Global Fault Address Register

The SMMU_*s*GFAR characteristics are:

Purpose Contains the input address of an erroneous request reported by SMMU_sGFSR.

Usage constraints See *Handling multiple memory faults on page 3-46* for information about when this register is updated. See also *Multiple faults on page 3-52*.

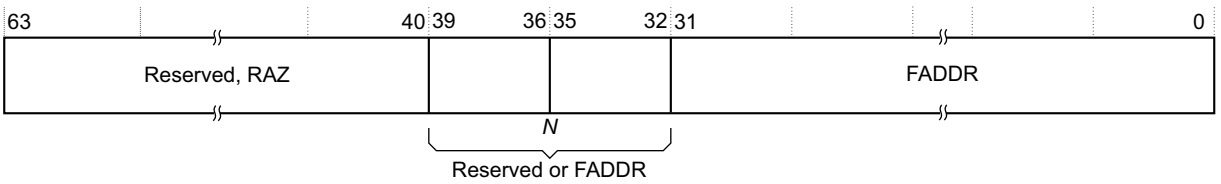
Configurations If an implementation includes 64-bit atomic access, this register can be accessed as a 64-bit quantity.

In an implementation that includes the Security Extensions, this register is Banked.

SMMU_NSGFAR is provided as a Secure alias of the Non-secure SMMU_GFAR. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 64-bit RW register with an UNKNOWN reset value.

The SMMU_*s*GFAR bit assignments are:



Bits[63:*N*] Reserved.

FADDR, bits[*N*-1:0]

Fault Address, the input address of the faulty access. For a Configuration access fault, this is the physical address resulting in the fault. For other fault classes, it is the input address of the faulting access, that the system can interpret in a number of ways.

Depending on the implemented input address space, the value of *N* can be:

- 32, for 4GB of address space.
- 36, for 64GB of address space.
- 40, for 1TB of address space.

SMMU_IDR2.IAS gives the implemented input address space.

10.6.15 SMMU_*s*GFSR, Global Fault Status Register

The SMMU_*s*GFSR characteristics are:

Purpose Gives the fault status for each of the following possible faults:

- Invalid context fault
- Unidentified stream fault
- Stream match conflict fault
- Unimplemented context bank fault
- Unimplemented context interrupt fault
- Configuration access fault
- External fault.

Usage constraints There are no usage constraints.

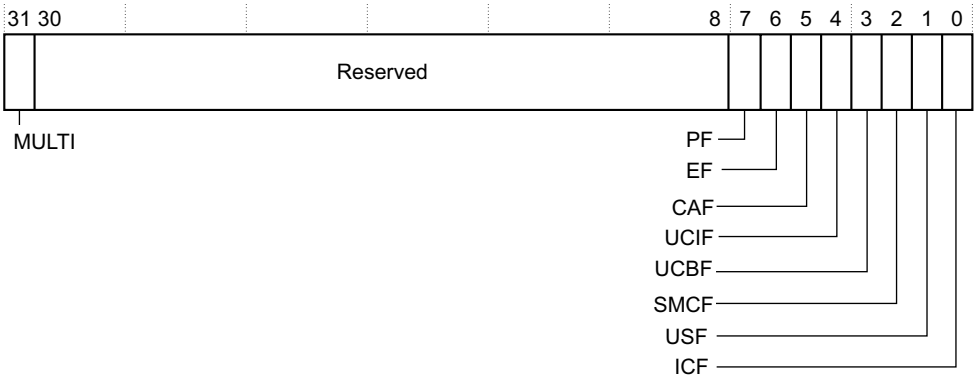
Configurations In an implementation that includes the Security Extensions, this register is Banked.

SMMU_NSGFSR is provided as a Secure alias of the Non-secure SMMU_GFSR. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit RW clear register. A value of 1 written to any non-reserved bit clears that bit. A value of 0 written to any of these bits leaves the bit unchanged.

This register resets to an UNKNOWN value.

The SMMU_*s*GFSR bit assignments are:



MULTI, bit[31] Multiple error condition. The possible values of this bit are:

0 No multiple error condition was encountered.

1 An error occurred while the value in SMMU_*s*GFSR was nonzero.

Bits[30:8] Reserved.

PF, bit[7] Permission Fault.

In SMMU_GFSR, this field is reserved.

In SMMU_SGFSR, this field records global SMMU_SCR1.SIF faults.

The possible values of this bit are:

0 No permission fault.

1 Permission fault.

Note

If a transaction is associated with a particular Translation context bank, faults are recorded in SMMU_CBn_FSR instead of SMMU_SGFSR.

EF, bit[6] External Fault. The possible values of this bit are:

0 No external fault.

1 External fault caused by an external abort.

CAF, bit[5] Configuration Access Fault. The possible values of this bit are:

0 No Configuration access fault.

1 Configuration access fault.

UCIF, bit[4] Unimplemented Context Interrupt Fault. The possible values of this bit are:

0 No Unimplemented context interrupt fault.

1 Unimplemented context interrupt fault.

UCBF, bit[3] Unimplemented Context Bank Fault. The possible values of this bit are:

0 No Unimplemented context bank fault.

1 Unimplemented context bank fault.

SMCF, bit[2] Stream Match Conflict Fault. The possible values of this bit are:

0 No Stream match conflict fault.

1 Stream match conflict fault.

USF, bit[1] Unidentified Stream Fault. The possible values of this bit are:

0 No Unidentified stream fault.

1 Unidentified stream fault.

ICF, bit[0] Invalid Context Fault. The possible values of this bit are:

0 Invalid context fault.

1 No Invalid context fault.

10.6.16 SMMU_*s*GFSRRESTORE, Global Fault Status Restore Register

The SMMU_*s*GFSRRESTORE characteristics are:

Purpose Restores the state of SMMU_sGFSR, after a reset, for example.

Usage constraints The SMMU_sGFSR register restored depends on the SMMU_*s*GFSRRESTORE register written to:

- writing to SMMU_GFSRRESTORE restores SMMU_GFSR
- writing to SMMU_SGFSRRESTORE restores SMMU_SGFSR.

Configurations If the implementation includes the Security Extensions, this register is Banked.

SMMU_NSGFSRRESTORE is provided as a Secure alias of the Non-secure SMMU_GFSRRESTORE. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit WO register.

The SMMU_*s*GFSRRESTORE bit assignments are identical to SMMU_sGFSR.

10.6.17 SMMU_*s*GFSYNR0, Global Fault Syndrome Register 0

The SMMU_*s*GFSYNR0 characteristics are:

Purpose Contains fault syndrome information relating to SMMU_sGFSR.

Usage constraints SMMU_GFSYNR0 is for Non-secure accesses. In an implementation that includes the Security Extensions, SMMU_SGFSYNR0 is for Secure accesses.

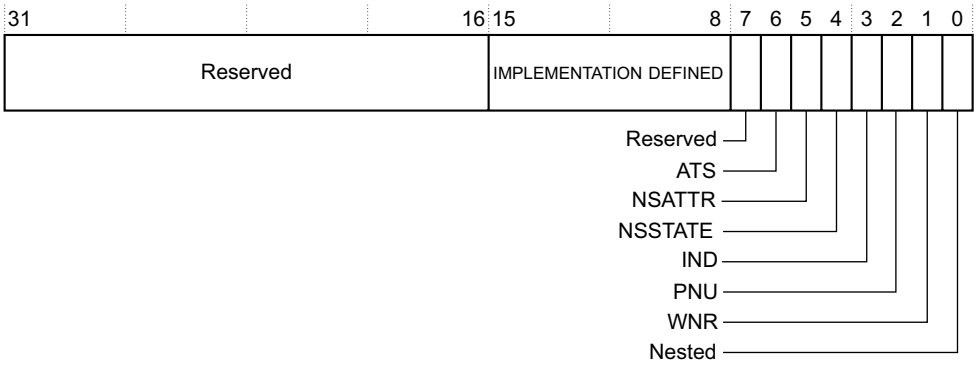
See *Multiple faults on page 3-52* for information about when this register is updated. See also *Multiple fault conditions on page 3-57*.

Configurations In an implementation that includes the Security Extensions, this register is Banked.

SMMU_NSGFSYNR0 is provided as a Secure alias of the Non-secure SMMU_GFSYNR0. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit RW register that resets to an UNKNOWN value.

The SMMU_*s*GFSYNR0 bit assignments are:



Bits[31:16] Reserved.

Bits[15:8] IMPLEMENTATION DEFINED.

Bit[7] Reserved.

ATS, Bit[6] Address translation operation fault. The possible values of this bit are:

0 The fault was not caused by the processing of an SMMU_CBn_ATS operation initiated in a Stage 1 Translation context bank.

1 The fault was caused by the processing of an SMMU_CBn_ATS operation initiated in a Stage 1 Translation context bank. See *Fault handling within nested translation operations initiated in a context bank on page 4-65* for more information.

In SMMU_SGFSYNR0 this field is reserved.

NSATTR, bit[5]

Non-Secure Attribute.The possible values of this bit are:

0 The faulty transaction has the Secure attribute.

1 The faulty transaction has the Non-secure attribute.

In SMMU_GFSYNR0 this field is reserved.

NSSTATE, bit[4]

Non-Secure State. The possible values of this bit are:

0 The faulty transaction is associated with a Secure device.

1 The faulty transaction is associated with a Non-secure device.

In SMMU_GFSYNR0 this field is reserved.

This field is set to 1 if a fault encountered when processing a Non-secure client transaction is reported to SMMU_SGFSR, for example, when:

- a Configuration access fault occurs as a response to a Non-secure attempt to access a Secure-only resource, such as SMMU_SCR1, or a context bank reserved for Secure software
- an external abort associated with Non-secure execution occurs, when the SMMU_SCR1.GEFRO bit is set to 1.

IND, bit[3] Instruction Not Data. The possible values of this bit are:

0 The faulty transaction has the data access attribute.

1 The faulty transaction has the instruction access attribute.

PNU, bit[2] Privileged Not Unprivileged. The possible values of this bit are:

0 The faulty transaction has the unprivileged attribute.

1 The faulty transaction is privileged.

WNR, bit[1] Write Not Read. The possible values of this bit are:

0 The faulty transaction is a read.

1 The faulty transaction is a write.

Nested, bit[0] Nested fault. The possible values of this bit are:

0 The fault occurred in the initial stream context.

1 The fault occurred in a nested context.

In SMMU_SGFSYNR0 this field is reserved.

10.6.18 SMMU_*s*GFSYNR1, Global Fault Syndrome Register 1

The SMMU_*s*GFSYNR1 characteristics are:

Purpose Contains fault syndrome information relating to SMMU_sGFSR.

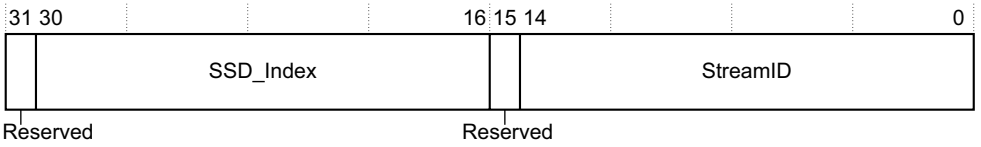
Usage constraints The security state of the faulty transaction determines whether SMMU_GFSYNR1 or SMMU_SGFSYNR1 is used.

Configurations In an implementation that includes the Security Extensions, this register is Banked.

SMMU_NSGFSYNR1 is provided as a Secure alias of the Non-secure SMMU_GFSYNR1. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit RW register that resets to an UNKNOWN value.

The SMMU_*s*GFSYNR1 bit assignments are:



Bit[31] Reserved.

SSD_Index, bits[30:16]

SSD_Index of the transaction that caused the fault.

The number of SSD_Index bits is IMPLEMENTATION DEFINED. Unimplemented bits behave as RAZ/WI.

The SSD_Index field is only accessible to configuration accesses by Secure software, using SMMU_NSGFSYNR1. Non-secure configuration accesses treat this field as RAZ/WI. This means that software must access SMMU_NSGFSYNR1 to obtain the SSD_Index for a faulty Non-secure transaction,

Bit[15] Reserved.

StreamID, bits[14:0]

StreamID of the transaction that caused the fault.

The number of StreamID bits is IMPLEMENTATION DEFINED. Unimplemented bits behave as RAZ/WI.

10.6.19 SMMU_*s*GFSYNR2, Global Fault Syndrome Register 2

The SMMU_*s*GFSYNR2 characteristics are:

Purpose Contains fault syndrome information relating to faults in SMMU_sGFSR.

Usage constraints Any usage constraints are IMPLEMENTATION DEFINED.

Configurations In an implementation that includes the Security Extensions, this register is Banked.

SMMU_NSGFSYNR2 is provided as a Secure alias of the Non-secure SMMU_GFSYNR2. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 32-bit RW register that resets to an UNKNOWN value.

The SMMU_*s*GFSYNR2 bit assignments are IMPLEMENTATION DEFINED.

10.6.20 SMMU_*s*GPAR, Global Physical Address Register

The SMMU_*s*GPAR characteristics are:

Purpose Holds the result of a global address translation operation.

Usage constraints See *Interaction with the Security Extensions on page 4-69*.

Configurations The format of this register depends on whether the Short-descriptor or Long-descriptor translation table format is selected, and whether the last global address translation operation was successful. See:

- *Short-descriptor translation table format*
- *Long-descriptor translation table format on page 10-137*
- *Fault format on page 10-137*.

See also *Multi-format registers and reserved fields on page 10-114*.

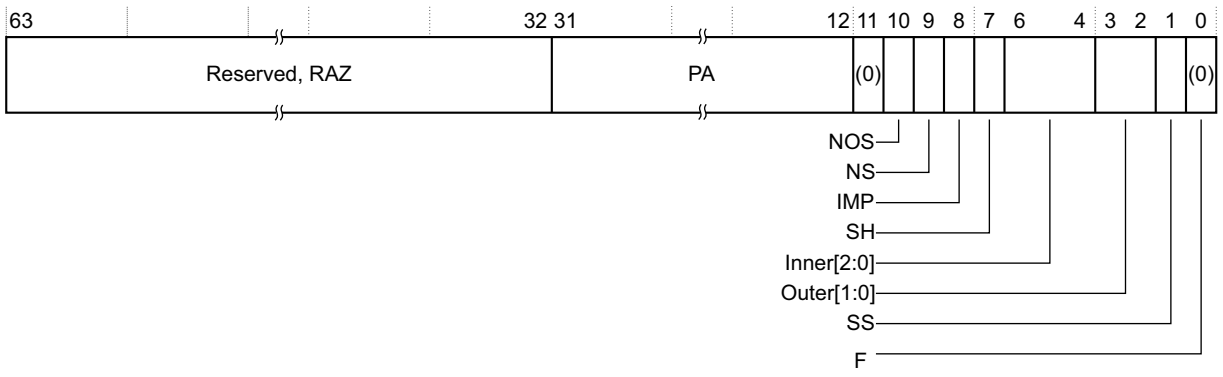
In an implementation that includes the Security Extensions, this register is Banked.

SMMU_NSGPAR is provided as a Secure alias of the Non-secure SMMU_GPAR. See Table 10-1 on page 10-104 and *Secure alias for Non-secure registers on page 10-111*.

Attributes A 64-bit RW register with an UNKNOWN reset value.

Short-descriptor translation table format

If the translation completes successfully, the format of the SMMU_*s*GPAR bit assignments is:



Bits[63:32] Reserved.

PA, bits[31:12] Bits[31:12] of the physical address.

NOS, bit[10] Not Outer Shareable attribute. Indicates whether the physical memory is Outer Shareable. The possible values of this bit are:

0 Memory is Outer Shareable.

1 Memory is not Outer Shareable.

Whether an implementation distinguishes between Inner Shareable and Outer Shareable memory is IMPLEMENTATION DEFINED. If an implementation does not make this distinction, NOS is UNK/SBZP.

NS, bit[9] Non-Secure. The NS attribute for a translation table entry read from a Secure Translation context bank.

This bit is UNKNOWN for a translation table entry read from a Non-secure Translation context bank.

In SMMU_GPAR this field is reserved.

IMP, bit[8] IMPLEMENTATION DEFINED.

SH, bit[7] Shareable attribute. Indicates whether the physical memory is shareable. The possible values of this bit are:

0 Physical memory is not shareable.

1 Physical memory is shareable.

Inner[2:0], bits[6:4]

Inner memory attributes from the translation table entry.

The encoding of this field is:

`0b111` Write-Back, no Write-Allocate.

`0b110` Write-Through.

`0b101` Write-Back, Write-Allocate.

`0b011` Device.

`0b001` Strongly-ordered.

`0b000` Non-cacheable.

All other values are reserved.

Outer[1:0], bits[3:2]

Outer memory attributes from the translation table. The encoding of this field is:

`0b00` Write-Back, no Write-Allocate.

`0b10` Write-Through, no Write-Allocate.

`0b01` Write-Back, Write-Allocate.

`0b11` Non-cacheable.

SS, bit[1] SuperSection, indicates whether the result is a supersection. The possible values of this bit are:

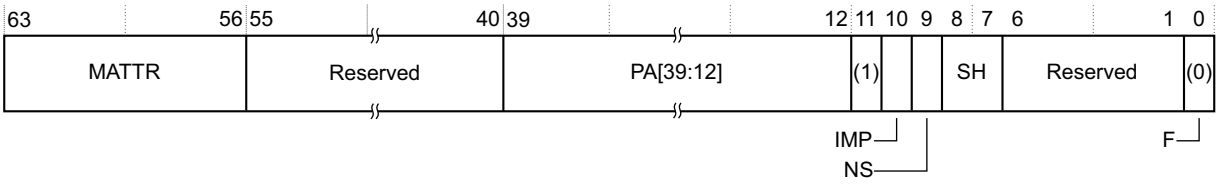
0 The page is not a supersection.
PAR[31:12] contains PAR[31:12] regardless of page size.

1 The page is part of a supersection.
PAR[31:24] contains PA[31:24].
PAR[23:16] contains PA[39:32].
PAR[15:12] contains `0b0000`.

F(0), bit[0] Fault, contains the value 0 on successful completion.

Long-descriptor translation table format

If a translation completes successfully, the format of the SMMU_*s*GPAR bit assignments is:



MATTR, bits[63:56]

Memory Attributes. These attributes have the encoding of the MAIR field. See *SMMU_CBn_MAIRm, Memory Attribute Indirection Registers on page 15-201* for more information.

Bits[55:40] Reserved.

PA[39:12], bits[39:12]

Bits[39:12] of the physical address.

IMP, bit[10] IMPLEMENTATION DEFINED.

NS, bit[9] Non-secure, the NS attribute for a translation table entry read from a Secure Translation context bank.

This bit is UNKNOWN for a translation table entry read from a Non-secure Translation context bank.

In SMMU_GPAR this field is reserved.

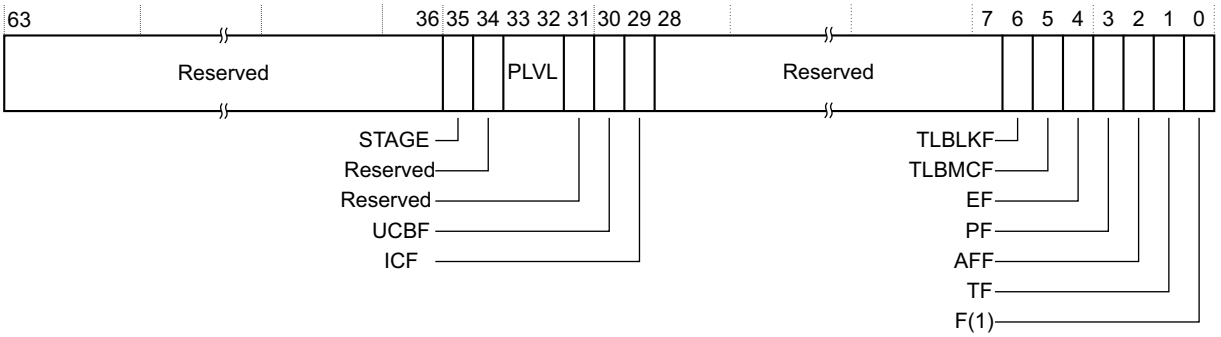
SH, bits[8:7] Shareability.

Bits[6:1] Reserved.

F(0), bit[0] Fault, contains the value 0 on successful completion.

Fault format

If a translation fails to complete successfully, the format of the SMMU_*s*GPAR bit assignments, irrespective of the translation format, is:



Bits[63:36] Reserved.

STAGE, bit[35]

The stage of translation that encountered the fault. The possible values of this bit are:

0 Fault encountered in stage 1 translation.

1 Fault encountered in stage 2 translation.

Bit[34] Reserved.

PLVL, bit[33:32]

Page Level.

Level of translation table walk that encountered the fault. The possible values of this bit are:

`0b01` Fault encountered in a level 1 translation table walk.

`0b10` Fault encountered in a level 2 translation table walk.

`0b11` Fault encountered in a level 3 translation table walk.

Bit[31] Reserved.

UCBF, bit[30] Unimplemented Context Bank Fault. The possible values of this bit are:

0 No fault.

1 Fault encountered because an unimplemented context bank was specified.

ICF, Bit[29] Invalid Context Fault. The possible values of this bit are:

0 No fault.

1 Fault encountered because an invalid context was specified during address translation.

This fault condition occurs if the stage 1 Translation context bank is associated with a stage 2 fault context in the SMMU_CBARn corresponding to the stage 1 context.

Bits[28:7] Reserved.

TLBLKF, bit[6]

TLB Lock Fault. The possible values of this bit are:

0 No fault.

1 TLB Lock fault.

TLBMCF, bit[5]

TLB Match Conflict Fault. The possible values of this bit are:

0 No fault.

1 Fault is a result of multiple matches detected in the TLB.

EF, bit[4] External Fault. The possible values of this bit are:

0 No fault.

1 An external fault.

PF, bit[3] Permission Fault. The possible values of this bit are:

0 No fault.

1 Fault caused by insufficient permission to complete access.

AFF, bit[2] Access Flag Fault. The possible values of this bit are:

0 No fault.

1 Access flag fault occurred.

TF, bit[1] Translation Fault. The possible values of this bit are:

0 No fault.

1 Invalid mapping for the address being accessed.

F(1), bit[0] Fault.

This bit is set to 1 if the translation aborts.

10.6.21 SMMU_*s*TLBGSTATUS, Global TLB Status register

The SMMU_*s*TLBGSTATUS characteristics are:

Purpose Gives the status of a TLB maintenance operation. See *Memory-mapped TLB maintenance operations on page 5-75* for more information.

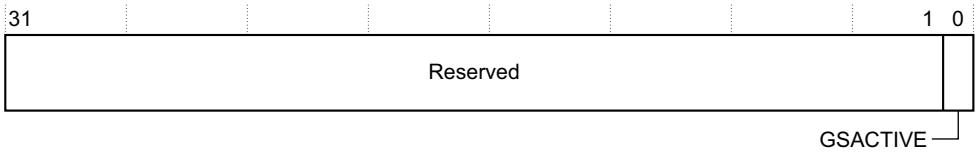
Usage constraints The security state of the transaction determines whether SMMU_TLBGSTATUS or SMMU_STLBGSTATUS is used.

Configurations In an implementation that includes the Security Extensions, this register is Banked.

SMMU_NSTLBGSTATUS is provided as a Secure alias of the Non-secure SMMU_TLBGSTATUS. See Table 10-1 on page 10-104. See also *Secure alias for Non-secure registers on page 10-111* for general information.

Attributes A 32-bit RO register.

The SMMU_*s*TLBGSTATUS bit assignments are:



Bits[31:1] Reserved.

GSACTIVE, bit[0]

Global Synchronize TLB Invalidate Active. The possible values of this bit are:

0 No Global TLB synchronization operation is active.

1 A Global TLB synchronization operation is active.

This bit resets to 0.

10.6.22 SMMU_*s*TLBGSYNC, Global Synchronize TLB Invalidate

The SMMU_*s*TLBGSYNC characteristics are:

Purpose Starts a global synchronization operation that ensures the completion of any previously accepted TLB Invalidate operation. As a minimum, the operation applies to the specified security state, and includes all TLB Invalidate operations initiated in context banks associated with that security state. See *Memory-mapped TLB maintenance operations on page 5-75* for more information.

Usage constraints The security state of the transaction determines whether SMMU_TLBGSYNC or SMMU_STLBGSYNC is used.

Configurations In an implementation that includes the Security Extensions:

- This register is Banked.
- SMMU_TLBGSYNC only has to apply to Non-secure TLB Invalidate operations.
- SMMU_STLBGSYNC only has to apply to Secure TLB Invalidate operations.

SMMU_NSTLBGSYNC is provided as a Secure alias of the Non-secure SMMU_TLBGSYNC. See Table 10-1 on page 10-104. See also *Secure alias for Non-secure registers on page 10-111* for general information.

Attributes A 32-bit WO command.

The SMMU_*s*TLBGSYNC bit assignments are reserved.

10.6.23 SMMU_S2CR*n*, Stream-to-Context Register

The SMMU_S2CR*n* characteristics are:

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

Usage constraints An SMMU_S2CR*n* register reserved by Secure software using SMMU_SCR1.NSNUMSMRGO must only specify a Translation context bank that is reserved by Secure software.

An SMMU_S2CR*n* register that is accessible from the Non-secure state must only specify a Translation context bank that is not reserved by Secure software.

Configurations The format of this register depends on the state of its TYPE[17:16] field. See:

- *Translation context, type 0b00*
- *Bypass mode, type 0b01 on page 10-142*
- *Fault context, type 0b10 on page 10-145*.

See also *Multi-format registers and reserved fields on page 10-114*.

The number of SMMU_S2CR*n* registers is IMPLEMENTATION DEFINED.

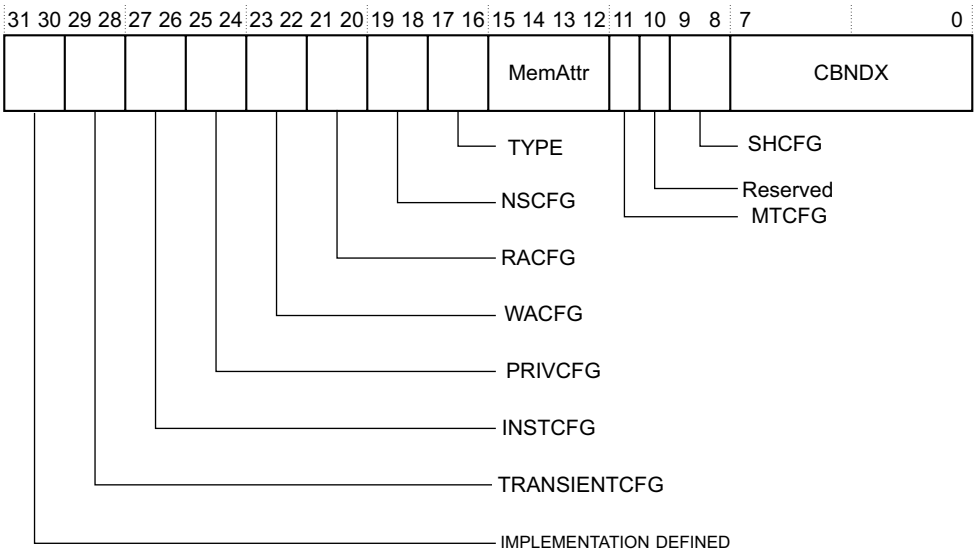
Unimplemented registers are RAZ/WI.

Attributes 32-bit RW registers with UNKNOWN reset values.

Translation context, type `0b00`

This register format specifies an initial Translation context bank.

The format of the bit assignments is:



Bits[31:30] IMPLEMENTATION DEFINED.

TRANSIENTCFG, bits[29:28]

Transient Allocate Configuration, controls the transient allocation hint.

The encoding of this field is:

`0b00` Use the default transient allocation attributes.

`0b01` Reserved.

`0b10` Non-transient.

`0b11` Transient.

It is IMPLEMENTATION DEFINED whether this field is present. If not present, these bits are RAZ/WI.

INSTCFG, bits[27:26]

Instruction Fetch Attribute Configuration. The encoding of this field is:

`0b00` Default instruction fetch attribute.

`0b01` Reserved.

`0b10` Data.

`0b11` Instruction.

It is IMPLEMENTATION DEFINED whether:

- this field transforms writes.
- the following transactions are restricted by Execute Never permission checking:
  - incoming write transactions already classified as Instruction
  - write transactions that INSTCFG encodes as Instruction

PRIVCFG, bits[25:24]

Privileged Attribute Configuration. The encoding of this field is:

`0b00` Default privilege attributes.

`0b01` Reserved.

`0b10` Unprivileged.

`0b11` Privileged.

WACFG, bits[23:22]

Write Allocation Configuration, controls the allocation hint for write accesses. The encoding of this field is:

`0b00` Default allocation attributes.

`0b01` Reserved.

`0b10` Write-Allocate.

`0b11` No Write-Allocate.

RACFG, bits[21:20]

Read Allocate Configuration, controls the allocation hint for read accesses. The encoding of this field is:

`0b00` Default allocation attributes.

`0b01` Reserved.

`0b10` Read-Allocate.

`0b11` No Read-Allocate.

NSCFG, bits[19:18]

Non-Secure Configuration. The encoding of this field is:

`0b00` Default security attribute.

`0b01` Reserved.

`0b10` Secure configuration that only affects Secure SMMU_S2CR*n* entry.

`0b11` Non-secure.

This field only exists for Secure Stream mapping register groups. For Non-secure Stream mapping register groups, it is reserved.

TYPE, bits[17:16]

Register type. Indicates the meaning of the remaining fields in this register. The encoding of this field is:

`0b00` Translation context bank index.

`0b01` Bypass mode.

`0b10` Fault, no index.

`0b11` Reserved.

MemAttr, bits[15:12]

Memory Attributes. See *Memory attribute, MemAttr on page 10-113*.

In an implementation that includes the Security Extensions, a Secure SMMU_S2CR*n* register configured to specify a Translation context bank is only permitted to:

- Specify a CBNDX corresponding to a Translation context bank that is also reserved by Secure software.
- Specify a Translation context bank configured for the *Stage 1 context with stage 2 bypass* format.

MTCFG, bit[11]

Memory Type Configuration. The possible values of this bit are:

0 Default memory attributes.

1 MemAttr field attributes.

Bit[10] Reserved.

SHCFG, bits[9:8]

Shared Configuration. The encoding of this field is:

`0b00` Default Shareable attribute.

`0b01` Outer Shareable.

`0b10` Inner Shareable.

`0b11` Non-shareable.

CBNDX, bits[7:0]

Context Bank Index. The Translation context bank index for a stage 1 or a stage 2 translation.

The number of CBNDX bits implemented is IMPLEMENTATION DEFINED. This field must be capable of selecting all of the implemented Translation context banks.

An implementation provides the same number of CBNDX bits for every implemented SMMU_S2CR*n*.

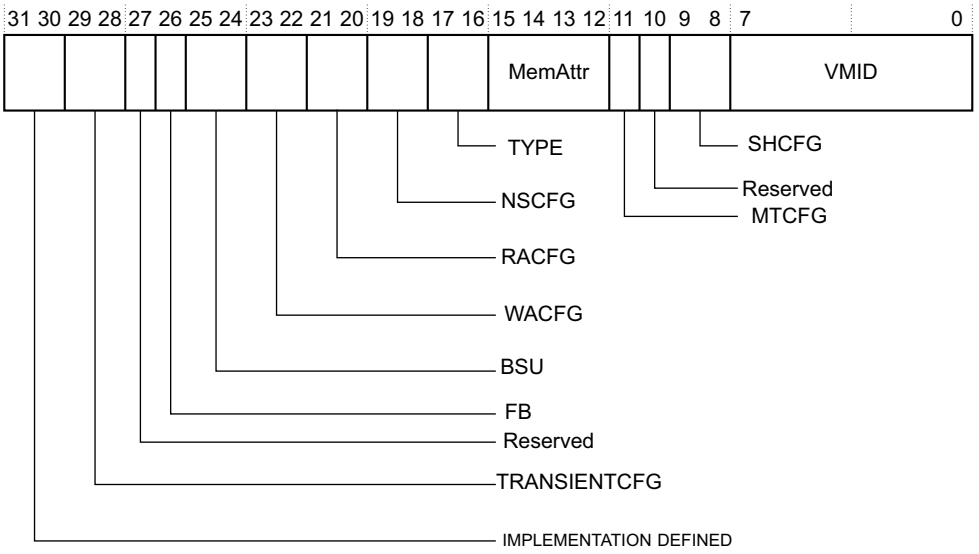
Unimplemented bits behave as RAZ/WI.

Bypass mode, type `0b01`

This register format specifies that Bypass mode is to be used.

No address translation is applied. Memory attributes can be specified to overlay or override those associated with the transaction.

The format of the bit assignments is:



Bits[31:30] IMPLEMENTATION DEFINED.

TRANSIENTCFG, bits[29:28]

Transient Allocate Configuration.

The encoding of this field is:

`0b00` Use default transient allocation attributes.

`0b01` Reserved.

`0b10` Non-transient.

`0b11` Transient.

It is IMPLEMENTATION DEFINED whether this field is present. If not present, it is RAZ/WI.

Bit[27] Reserved.

FB, bits[26] Force Broadcast. Force Broadcast of TLB, branch predictor and instruction cache maintenance operations.

This field affects client TLB maintenance, BPIALL and ICIALLU operations. If it has the value 1, any affected operation is modified to the equivalent broadcast variant within 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.

BSU, bits[25:24]

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 applied to any barrier.

The encoding of this field is:

`0b00` No effect.

`0b01` Inner Shareable.

`0b10` Outer Shareable.

`0b11` Full system.

Upgrade of the barrier shareability domain might not be supported in all system topologies. In an implementation that does not have this upgrade behavior, this field is RAZ/SBZP.

WACFG, bits[23:22]

Write-Allocate Configuration, an allocation hint for write accesses. The encoding of this field is:

`0b00` Default allocation attributes.

`0b01` Reserved.

`0b10` Write-Allocate.

`0b11` No Write-Allocate.

RACFG, bits[21:20]

Read-Allocate Configuration. Gives an allocation hint for read accesses. The encoding of this field is:

`0b00` Default allocation attributes.

`0b01` Reserved.

`0b10` Read-Allocate.

`0b11` No Read-Allocate.

NSCFG, bits[19:18]

Non-Secure Configuration. The encoding of this field is:

`0b00` Default security attribute.

`0b01` Reserved.

`0b10` Secure configuration that only affects Secure SMMU_S2CR*n* entry.

`0b11` Non-secure.

This field only exists for Secure Stream mapping register groups. For Non-secure Stream mapping register groups, it is reserved.

TYPE, bits[17:16]

Register type. Indicates the meaning of the remaining fields in this register. The encoding of this field is:

`0b00` Translation context bank index.

`0b01` Bypass mode.

`0b10` Fault, no index.

`0b11` Reserved.

MemAttr, bits[15:12]

Memory Attributes. See *Memory attribute, MemAttr on page 10-113*.

In an implementation that includes the Security Extensions, a Secure SMMU_S2CR*n* register configured to specify a Translation context bank is only permitted to:

- specify a CBNDX corresponding to a Translation context bank that is also reserved by Secure software
- specify a Translation context bank configured to the *Stage 1 context with stage 2 bypass* format.

MTCFG, bit[11]

Memory Type Configuration. The possible values of this bit are:

0 Default memory attributes.

1 MemAttr field attributes.

Bit[10] Reserved.

SHCFG, bits[9:8]

Shared Configuration. The encoding of this field is:

`0b00` Default Shareable attribute.

`0b01` Outer Shareable.

`0b10` Inner Shareable.

`0b11` Non-shareable.

VMID, bits[7:0]

Reserved for IMPLEMENTATION DEFINED use of a VMID field, where such use is relevant to all transactions, including those not subject to any translation.

It is IMPLEMENTATION DEFINED whether this field is implemented, and what effect it has.

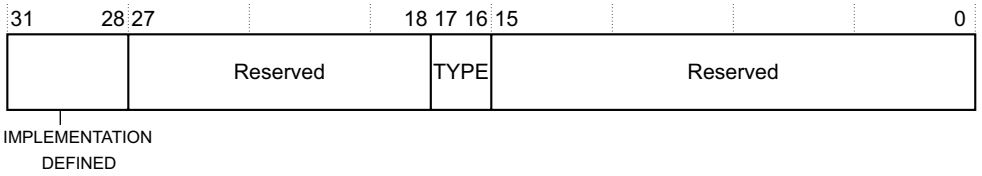
In an implementation that does not include this behavior, this field is RAZ/WI.

If an implementation that includes the Security Extensions does not support a VMID for Secure translation regimes, any Secure VMID field is ignored. See *Interaction with the Security Extensions on page 2-32* for more information.

Fault context, type `0b10`

This format specifies that the Fault context is to be used. Any transaction that maps to this Stream mapping group incurs an Invalid context fault.

The format of the bit assignments is:



Bits[31:28] IMPLEMENTATION DEFINED.

Bits[27:18] Reserved.

TYPE, bits[17:16]

Register Type. Indicates the meaning of the remaining fields in this register. The encoding of this field is:

`0b00` Translation context bank index.

`0b01` Bypass mode.

`0b10` Fault, no index.

`0b11` Reserved.

Bits[15:0] Reserved.

10.6.24 SMMU_SCR1, Secure Configuration Register 1

The SMMU_SCR1 characteristics are:

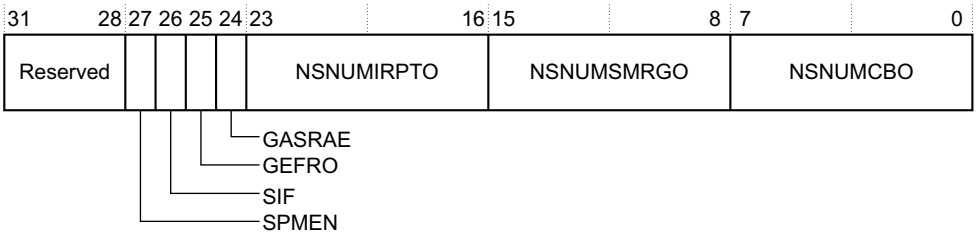
Purpose Provides top-level Secure control of the System MMU.

Usage constraints Secure access only.

Configurations Only present in an implementation that includes the Security Extensions.

Attributes A 32-bit RW register. See the field descriptions for information about the reset values.

The SMMU_SCR1 bit assignments are:



Bits[31:28] Reserved.

SPMEN, bit[27]

Secure Performance Monitor Enable. The possible values of this bit are:

0 Any event caused by Secure transaction processing does not contribute towards performance monitor counting.

1 Any events caused by Secure transaction processing is permitted to contribute towards performance monitor counting.

This bit resets to 0.

SIF, bit[26] Secure Instruction Fetch. The possible values of this bit are:

0 Secure instruction fetches are permitted to Non-secure memory locations.

1 Raise a permission fault if a Secure domain access attempts to exit the SMMU as a Non-secure instruction.

See *Secure Instruction Fetch (SIF) permission faults on page 3-51* for more information.

This bit resets to 0.

Note

If a transaction is associated with a particular Translation context bank, faults are recorded in SMMU_CBn_FSR. Otherwise they are recorded in SMMU_SGFSR.

GEFRO, bit[25]

Global External Fault Report Override. The possible values of this bit are:

0 Permit SMMU_GFSR to report external faults.

1 SMMU_SGFSR reports all external faults.

If SMMU_SCR1.GEFRO==1, all external aborts that would have been recorded in SMMU_GFSR are instead recorded in SMMU_SGFSR. See *SMMU_sGFSR, Global Fault Status Register on page 10-131*.

This bit resets to an UNKNOWN value.

GASRAE, bit[24]

Global Address Space Restricted Access Enable. The possible values of this bit are:

0 The Global address space has default access permission, permitting Secure and Non-secure configuration memory accesses.

1 The Global address space is only accessible by Secure configuration memory accesses. Stage 2 format context banks are only accessible by Secure configuration accesses.

Whether SMMU_SCR1.GASRAE==1 affects the IMPLEMENTATION DEFINED address space is IMPLEMENTATION DEFINED.

The following additional constraints apply:

- If SMMU_SCR1.GASRAE==0, Secure software must avoid setting SMMU_CBARn.HYPC to 1 when configuring a Secure Stage 1 Translation context bank.
- If SMMU_SCR1.GASRAE==1, Secure software must avoid setting SMMU_CBARn.HYPC to 1 when configuring a Non-secure Stage 1 Translation context bank.

UNPREDICTABLE behavior results if Secure software does not follow these constraints.

This bit resets to 0.

NSNUMIRPTO, bits[23:16]

Non-Secure Number of Interrupts Override.

It is IMPLEMENTATION DEFINED whether this field is fully implemented with writable storage. A portion of this field might behave as RAZ/WI because the implementation might not provide a sufficient number of interrupts for some of the upper bits of NSNUMIRPTO to be relevant.

In implementations that do not support Secure translation contexts, such as those that support stage 2 translation format only, it is also IMPLEMENTATION DEFINED whether this field is read-only. In such implementations, Secure software does not manage translation contexts and therefore there is no requirement to reserve context interrupts for Secure use.

These bits reset to the implemented number of interrupts. See *SMMU_IDR0-7, Identification registers on page 10-116*.

For more information, see:

- *Resource reservation on page 8-92*
- *SMMU_IDR0 on page 10-116*.

NSNUMSMRGO, bits[15:8]

Non-Secure Number of Stream Mapping Register Groups Override.

Adjusts the number of Stream mapping register groups visible to Non-secure accesses. The number of Stream mapping register groups reported in SMMU_IDR0 is reduced to the number indicated by SMMU_SCR1.NSNUMSMRGO. See *SMMU_IDR0-7, Identification registers on page 10-116*.

These bits reset to the implemented number of Stream mapping register groups. See *SMMU_IDR0-7, Identification registers on page 10-116*.

It is IMPLEMENTATION DEFINED whether this field is fully implemented with writable storage. A portion of this field might be RAZ/WI because the implementation might not provide sufficient Stream mapping register groups for some of the upper bits of SMMU_SCR1.NSNUMSMRGO to be relevant.

For more information, see:

- *The Stream mapping table on page 2-25*
- *Resource reservation on page 8-92*
- *Permitted transaction resource usage on page 8-92*
- *SMMU_IDR0 on page 10-116*.

NSNUMCBO, bits[7:0]

Non-Secure Number of Context Banks Override.

Adjusts the number of Translation context banks visible to Non-secure accesses. The number of Translation context banks reported in SMMU_IDR1.NUMCB is reduced to the number indicated by SMMU_SCR1.NSNUMCBO. See *SMMU_IDR0-7, Identification registers on page 10-116*.

SMMU_SCR1.NSNUMCBO also reserves the SMMU_CBARn registers and SMMU_CBFRSYNRAn registers associated with the reserved Translation context banks.

This field resets to the physically implemented number of Translation context banks.

This field must not be set to a value below that reported by SMMU_IDR1.NUMS2CB. Stage 2 format translation is not available to Secure transactions.

It is IMPLEMENTATION DEFINED whether this field is fully implemented with writable storage. A portion of this field might behave as RAZ/WI. This is permitted because the implementation might not provide sufficient contexts for some of the upper bits of SMMU_SCR1.NSNUMCBO to be relevant.

In implementations that do not support Secure translation contexts, such as those that support stage 2 translation format only, it is also IMPLEMENTATION DEFINED whether this field is read-only. In such implementations, Secure software does not manage translation contexts and there is therefore no requirement to reserve context interrupts for Secure use.

For more information, see:

- *The Translation context bank table on page 2-30*
- *The Context Bank Attribute Register, SMMU_CBARn on page 2-32*
- *Recording a context fault on page 3-51*
- *Resource reservation on page 8-92*
- *SMMU_IDR1 on page 10-117*.

10.6.25 SMMU_SMR*n*, Stream Match Register

The SMMU_SMR*n* characteristics are:

Purpose Match a transaction with a particular Stream mapping register group.

Usage constraints During configuration, the Stream Match Register table can have multiple entries that match the same Stream Identifier value, possibly resulting in UNPREDICTABLE behavior. See *StreamID matching on page 2-25* for more information about multiple matches. To prevent multiple matches, software must ensure that no transactions that might match the StreamID are received, by:

- disabling any source client devices that might match
- ensuring that no outstanding transactions from these client devices are in progress.

As an extra precaution, software can first disable all affected SMMU_SMR*n* table entries by setting the SMMU_SMR*n*.VALID bit to 1, then reprogramming the entries as appropriate.

An implementation must provide the same number of ID and MASK bits for every implemented Stream Match Register.

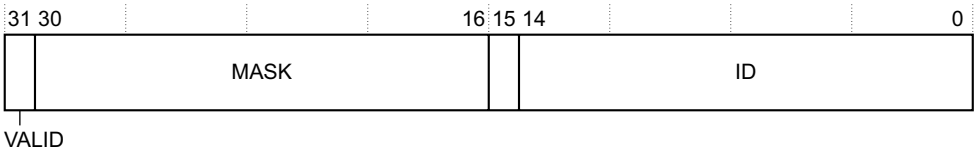
Configurations The number of SMMU_SMR*n* registers is IMPLEMENTATION DEFINED.

In an implementation that includes StreamID matching, each of these registers has RW access. In an implementation that includes StreamID indexing, each register is RAZ/WI.

Attributes 32-bit RW registers with access attributes that depend on the configuration of the implementation. See the configuration details for information.

This register resets to an UNKNOWN value.

The SMMU_SMR*n* bit assignments are:



VALID, bit[31]

The possible values of this bit are:

0 Entry is included in the Stream mapping table search.

1 Entry is not included in the Stream mapping table search.

MASK, bits[30:16]

Masking of StreamID bits irrelevant to the matching process:

- If MASK[i]==1, ID[i] is ignored.
- If MASK[i]==0, ID[i] is relevant for matching.

The actual number of MASK bits is IMPLEMENTATION DEFINED. Unimplemented bits are RAZ/WI.

Bit[15] Reserved.

ID, bits[14:0] The Stream Identifier to match.

The actual number of ID bits is IMPLEMENTATION DEFINED. Unimplemented bits are RAZ/WI.

10.6.26 SMMU_STLBIALL, TLB Invalidate All

The SMMU_STLBIALL characteristics are:

Purpose Invalidates all unlocked Secure entries in the TLB. See *Memory-mapped TLB maintenance operations on page 5-75* for more information.

Usage constraints This command must apply to all unlocked Hyp-tagged entries. Optionally, it can apply to other individual unlocked entries, regardless of their tagging.

Configurations In an implementation that does not include the Security Extensions, this command is reserved.

Attributes A 32-bit WO command.

The SMMU_STLBIALL bit assignments are reserved.

10.6.27 SMMU_TLBIALLH, TLB Invalidate All Hyp

The SMMU_TLBIALLH characteristics are:

Purpose Invalidates all Hyp tagged entries in the TLB. See *Memory-mapped TLB maintenance operations on page 5-75* for more information.

Usage constraints This command must apply to all unlocked Hyp-tagged entries. Optionally, it can apply to other individual unlocked entries, regardless of their tagging.

Configurations None.

Attributes A 32-bit WO command.

The SMMU_TLBIALLH bit assignments are reserved.

10.6.28 SMMU_TLBIALLNSNH, TLB Invalidate All Non-Secure Non-Hyp

The SMMU_TLBIALLNSNH characteristics are:

Purpose Invalidates all Non-secure non-Hyp tagged entries in the TLB. See *Memory-mapped TLB maintenance operations on page 5-75* for more information.

Usage constraints This command must apply to all unlocked Non-secure non-Hyp tagged entries. Optionally, it can apply to other individual unlocked entries, regardless of their tagging.

Configurations None.

Attributes A 32-bit WO command.

The SMMU_TLBIALLNSNH bit assignments are reserved.

10.6.29 SMMU_TLBIVAH, Invalidate Hyp TLB by VA

The SMMU_TLBIVAH characteristics are:

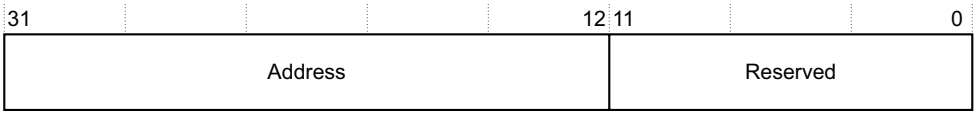
Purpose Invalidates all Hyp TLB entries that match the specified virtual address. See *Memory-mapped TLB maintenance operations on page 5-75* for more information.

Usage constraints This command must operate on all unlocked Hyp-tagged TLB entries associated with the specified virtual address. Optionally, it can apply to other individual unlocked entries in a TLB, regardless of their tagging.

Configurations None.

Attributes A 32-bit WO command.

The SMMU_TLBIVAH bit assignments are:



Address, bits[31:12] The virtual address to use.

Bits[11:0] Reserved.

10.6.30 SMMU_TLBIVMID, TLB Invalidate by VMID

The SMMU_TLBIVMID characteristics are:

Purpose Invalidates all Non-secure non-Hyp TLB entries having the specified VMID. See *Memory-mapped TLB maintenance operations on page 5-75* for more information.

Usage constraints This command must operate on all unlocked Non-secure non-Hyp TLB entries associated with the specified VMID. Optionally, it can apply to other individual unlocked entries in a TLB, regardless of their tagging.

This command does not affect Secure TLB entries.

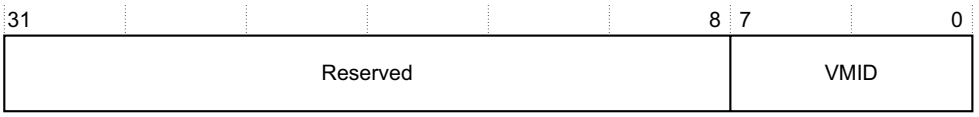
Note

In implementations that support a VMID for Secure context banks, TLB entries created by Secure banks are not tagged with the VMID. This command does not affect such entries. See *Interaction with the Security Extensions on page 2-32* for more information.

Configurations None.

Attributes A 32-bit WO command.

The SMMU_*s*TLBIVMID bit assignments are:



Bits[31:8] Reserved.

VMID, bits[7:0] The Virtual Machine Identifier to use in the Invalidate operation.

Chapter 11
System MMU Global Register Space 1

This chapter defines System MMU Global Register Space 1. It contains the following sections:

11.1 System MMU Global Register Space 1 register summary

System MMU Global Register Space 1 provides high-level control of System MMU resources.

The size of System MMU Global Register Space 1 is defined by *PAGESIZE*. See *PAGESIZE and NUMPAGENDXB on page 9-98*.

Table 11-1 shows the register space relative to the offset from *SMMU_GR1_BASE*.

Table 11-1 System MMU Global Register Space 1

Offset	Name	Type	Description
0x00000	SMMU_CBAR0	RW	*SMMU_CBARn, Context Bank Attribute Registers on page 11-155*
0x00004	SMMU_CBAR1		
0x00008-0x001FC	SMMU_CBAR2 to SMMU_CBAR127		
0x00200-0x003FC	Reserved	-	-
0x00400	SMMU_CBFRSYNRA0	RW	*SMMU_CBFRSYNRAn, Context Bank Fault Restricted Syndrome Register A on page 11-161*
0x00404	SMMU_CBFRSYNRA1		
0x00408-0x005FC	SMMU_CBFRSYNRA2 to SMMU_CBFRSYNRA127		
0x00600-(*PAGESIZE* − 0x4)	Reserved	-	-

11.2 System MMU Global Register Space 1 register descriptions

This section describes all of the Global Register Space 1 registers that might be present in a System MMU implementation. Registers are shown in register name order.

11.2.1 SMMU_CBAR*n*, Context Bank Attribute Registers

The SMMU_CBAR*n* characteristics are:

Purpose Specifies configuration attributes for Translation context bank *n*.

Usage constraints

If SMMU_IDR0.NTS is 0, nested translation is not supported, and using *Stage 1 context with stage 2 context, type 0b11 on page 11-159* generates an invalid context fault.

SMMU_IDR1.NUMS2CB specifies the IMPLEMENTATION DEFINED number of Translation context banks that only support stage 2 translation. See *The Translation context bank table on page 2-30* for more information about discovering whether an implementation uses such context banks.

In an implementation that includes the Security Extensions, Secure context banks must only use *Stage 1 context with stage 2 bypass, type 0b01 on page 11-156*. Using any other type generates an invalid context fault.

Configurations

The format of this register depends on the state of its TYPE field. See:

- *Stage 2 context, type 0b00*
- *Stage 1 context with stage 2 bypass, type 0b01 on page 11-156*
- *Stage 1 context with stage 2 fault, type 0b10 on page 11-158*
- *Stage 1 context with stage 2 context, type 0b11 on page 11-159*.

An SMMU implementation can configure the TYPE field to be read-only. For example, in an implementation that supports only stage 2 translation, this could hold a read-only value of `0b00`, removing any requirement to implement storage for this field.

The number of SMMU_CBAR*n* registers is IMPLEMENTATION DEFINED.

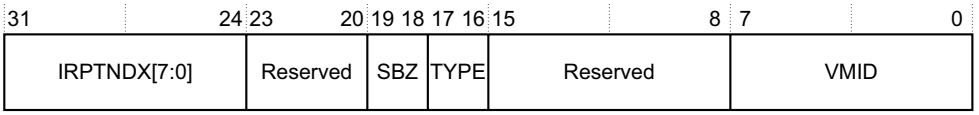
See also *Multi-format registers and reserved fields on page 10-114*.

Attributes 32-bit RW registers.

Stage 2 context, type `0b00`

This format configures the associated Translation context bank to provide only stage 2 translation.

The format of the bit assignments is:



IRPTNDX[7:0], bits[31:24]

Interrupt Index. The Context interrupt number to assert in the event of an interrupt raising a fault in the associated Translation context bank.

SMMU_IDR0.NUMIRPT specifies the range of values that software can configure this field in.

It is IMPLEMENTATION DEFINED whether this field is fully implemented with writable storage. A portion of this field is permitted to behave as RAZ/WI because the System MMU implementation might not provide sufficient interrupts for some of the upper bits of SMMU_CBAR*n*.IRPTNDX to be relevant. The implemented range of this field is the same for all SMMU_CBAR*n* registers in an implementation.

Bits[23:20] Reserved.

SBZ, bits[19:18]

Should-Be-Zero.

TYPE, bits[17:16]

Register Type, indicates the format of the remaining fields in this register. The encoding of this field is:

`0b00` Stage 2 context.

`0b01` Stage 1 context with stage 2 bypass.

`0b10` Stage 1 context with stage 2 fault.

`0b11` Stage 1 context with stage 2 context, that is, nested translation.

Bits[15:8] Reserved.

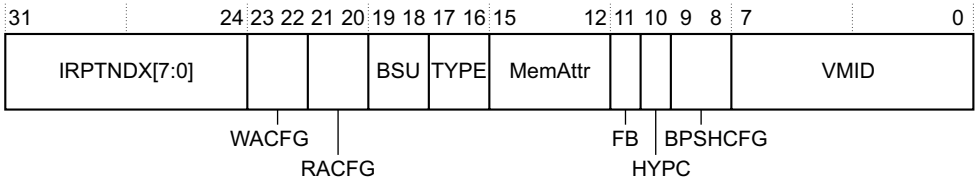
VMID, bits[7:0]

The Virtual Machine Identifier to be associated with the Translation context bank.

Stage 1 context with stage 2 bypass, type `0b01`

This format configures the associated Translation context bank to provide stage 1 translation, with Bypass mode instead of a second translation context.

The format of the bit assignments is:



IRPTNDX[7:0], bits[31:24]

Interrupt Index. The Context interrupt number to assert in the event of an interrupt raising a fault in the associated Translation context bank.

SMMU_IDR0.NUMIRPT specifies the range of values that software can configure this field in.

It is IMPLEMENTATION DEFINED whether this field is fully implemented with writable storage. A portion of this field is permitted to behave as RAZ/WI. This is permitted because the implementation might not provide sufficient interrupts for some of the upper bits of SMMU_CBAR*n*.IRPTNDX to be relevant. The implemented range of this field is the same for all SMMU_CBAR*n* registers in an implementation.

WACFG, bits[23:22]

Write-Allocate Configuration, allocation hint for write accesses. The encoding of this field is:

`0b00` Default allocation attributes.

`0b01` Reserved.

`0b10` Write-Allocate.

`0b11` No Write-Allocate.

RACFG, bits[21:20]

Read-Allocate Configuration, allocation hint for read accesses. The encoding of this field is:

`0b00` Default allocation attributes.

`0b01` Reserved.

`0b10` Read-Allocate.

`0b11` No Read-Allocate.

BSU, bits[19:18]

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 encoding of this field is:

0b00 No effect.

0b01 Inner Shareable.

0b10 Outer Shareable.

0b11 Full system.

Upgrade of the barrier shareability domain might not be supported in all system topologies. In an implementation that does not include this upgrade behavior, BSU is RAZ/SBZP.

TYPE, bits[17:16]

Register Type. Indicates the format of the remaining fields in this register. The encoding of this field is:

0b00 Stage 2 context.

0b01 Stage 1 context with stage 2 bypass.

0b10 Stage 1 context with stage 2 fault.

0b11 Stage 1 context with stage 2 nested.

MemAttr, bits[15:12]

Memory Attributes. See *Memory attribute, MemAttr on page 10-113*.

This field is combined with the shared attributes of the previous translation stage. See Table 11-2 on page 11-158.

FB, bit[11] Force Broadcast.

Force Broadcast of TLB, branch predictor and instruction cache maintenance operations.

This field affects client TLB maintenance, BPIALL and ICIALLU operations. If this field is 1, the affected operation is modified to the equivalent broadcast variant in the Inner Shareable domain.

The possible values of this bit are:

0 Process the affected operations as presented.

1 Upgrade the affected operations to be broadcast in the Inner Shareable domain.

HYPC, bit[10] Hypervisor Context. The possible values of this bit are:

0 Non-hypervisor context. Use VMID and ASID for TLB tagging.

1 Hypervisor context. Do not use VMID and ASID for TLB tagging.

In an interaction with the Security Extensions, the following restrictions apply to Secure software:

- If SMMU_SCR1.GASRAE==0, Secure software must not set HYPC to 1 for any Secure Translation context bank.
- If SMMU_SCR1.GASRAE==1, Secure software must not set HYPC to 1 for any Non-secure Translation context bank.

Otherwise, UNPREDICTABLE behavior might occur.

See *Hypervisor-marked contexts on page 2-41* for more information.

BPSHCFG, bits[9:8]

Bypass Shared Configuration. The encoding of this field is:

0b00 Reserved.

0b01 Outer Shareable.

0b10 Inner Shareable.

0b11 Non-shareable.

This field is combined with the shared attributes of the previous translation stage, as shown in Table 11-2.

Table 11-2 Shared attribute combination results

Stage 1 shareability	Stage 2 shareability	Resulting shareability
Outer Shareable	-	Outer Shareable
Inner Shareable	Outer Shareable	Outer Shareable
Inner Shareable	Inner Shareable	Inner Shareable
Inner Shareable	Non-Shareable	Inner Shareable
Non-Shareable	Outer Shareable	Outer Shareable
Non-Shareable	Outer Shareable	Outer Shareable
Non-Shareable	Inner Shareable	Inner Shareable
Non-Shareable	Non-Shareable	Non-Shareable

VMID, bits[7:0]

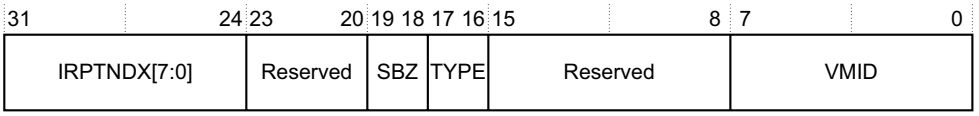
The Virtual Machine Identifier to be associated with the Translation context bank.

If an implementation that includes the Security Extensions does not support a VMID for Secure translation regimes, any Secure VMID field is ignored. See *Interaction with the Security Extensions on page 2-32* for more information.

Stage 1 context with stage 2 fault, type `0b10`

This format configures the associated Translation context bank to provide stage 1 translation, additionally specifying the fault context instead of a stage 2 translation context. Any transaction that maps to the corresponding Translation context bank incurs an Invalid context fault.

The format of the bit assignments is:



IRPTNDX[7:0], bits[31:24]

Interrupt Index. The Context interrupt number to assert in the event of an interrupt raising a fault in the associated Translation context bank.

SMMU_IDR0.NUMIRPT specifies the range of values that software can configure this field in.

It is IMPLEMENTATION DEFINED whether this field is fully implemented with writable storage. A portion of this field is permitted to behave as RAZ/WI, because the implementation might not provide sufficient interrupts for some of the upper bits of SMMU_CBAR*n*.IRPTNDX to be relevant. The implemented range of this field is the same for all SMMU_CBAR*n* registers in an implementation.

Bits[23:20] Reserved.

SBZ, bits[19:18]

Should-Be-Zero.

TYPE, bits[17:16]

Register Type. Indicates the format of the remaining fields in this register. The encoding of this field is:

`0b00` Stage 2 context.

`0b01` Stage 1 context with stage 2 bypass.

`0b10` Stage 1 context with stage 2 fault.

`0b11` Stage 1 context with stage 2 nested context.

Bits[15:8] Reserved.

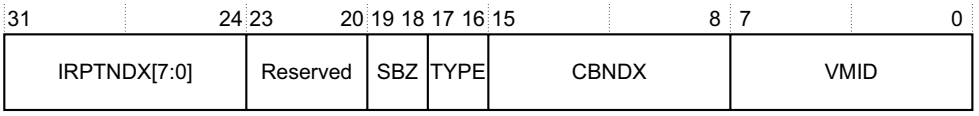
VMID, bits[7:0]

The Virtual Machine Identifier to be associated with the Translation context bank.

Stage 1 context with stage 2 context, type `0b11`

This format configures the associated Translation context bank to provide stage 1 translation, and specifies an additional Translation context bank as the Stage 2 Translation context bank in a nested translation.

The format of the bit assignments is:



IRPTNDX[7:0], bits[31:24]

Interrupt Index. The Context interrupt number to assert in the event of an interrupt raising a fault in the associated Translation context bank.

SMMU_IDR0.NUMIRPT specifies the range of values that software must configure this field in.

It is IMPLEMENTATION DEFINED whether this field is fully implemented with writable storage. A portion of this field is permitted to behave as RAZ/WI, because the implementation might not provide sufficient interrupts for some of the upper bits of SMMU_CBAR*n*.IRPTNDX to be relevant. The implemented range of this field is the same for all SMMU_CBAR*n* registers in an implementation.

Bits[23:20] Reserved.

SBZ, bits[19:18] Should-Be-Zero.

TYPE, bits[17:16]

Register Type. Indicates the format of the remaining fields in this register. The encoding of this field is:

`0b00` Stage 2 context.

`0b01` Stage 1 context with stage 2 bypass.

`0b10` Stage 1 context with stage 2 fault.

`0b11` Stage 1 context with nested stage 2 context.

CBNDX, bits[15:8]

Context Bank Index. The Translation context bank index used for the Stage 2 Translation context bank in a nested translation.

Behavior is UNPREDICTABLE if the SMMU_CBAR*n* register associated with the translation context specified by SMMU_CBAR*n*.CBNDX has any value other than `0b00` to specify a Stage 2 Translation context bank.

It is IMPLEMENTATION DEFINED whether this field is fully implemented with writable storage. A portion of this field is permitted to behave as RAZ/WI, because the implementation might not support a sufficient number of Translation context banks for some of the upper bits of SMMU_CBAR*n*.CBNDX to be relevant. The implemented range of this field is the same for all SMMU_CBAR*n* registers in an implementation.

VMID, bits[7:0]

The Virtual Machine Identifier to be associated with the Translation context bank.

11.2.2 SMMU_CBFRSYNRA*n*, Context Bank Fault Restricted Syndrome Register A

The SMMU_CBFRSYNRA*n* characteristics are:

Purpose Gives fault syndrome information about the access that caused an exception in the associated Translation context bank.

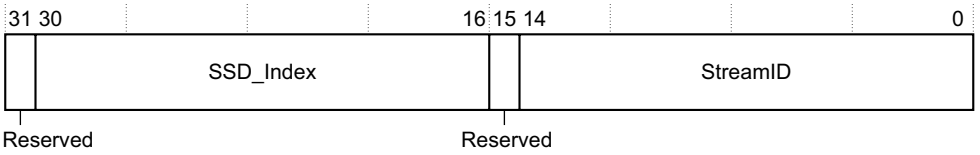
Usage constraints The value of this register is UNKNOWN if the recorded fault was an address translation fault, that is, if the SMMU_CB*n*_FSYNR0.ATOF bit is set to 1.

Configurations The number of SMMU_CBFRSYNRA registers is IMPLEMENTATION DEFINED. See also SMMU_IDR1.NUMCB.

For more information about SMMU_CBFRSYNRA*n*. See *Context Bank Fault Restricted Syndrome Register, SMMU_CBFRSYNRAn on page 3-52*.

Attributes 32-bit RW registers.

The format of the bit assignments is:



Bit[31] Reserved.

SSD_Index, bits[30:16]

The SSD_Index of the transaction that caused the fault.

The number of SSD_Index bits is IMPLEMENTATION DEFINED. Unimplemented bits behave as RAZ/WI.

This field is only accessible to configuration accesses by Secure software. Non-secure configuration accesses treat this field as RAZ/WI.

Bit[15] Reserved.

StreamID, bits[14:0]

The StreamID of the transaction that caused the fault.

The number of StreamID bits is IMPLEMENTATION DEFINED. Unimplemented bits behave as RAZ/WI.

Chapter 12
System MMU IMPLEMENTATION DEFINED Address Space

This chapter specifies the System MMU IMPLEMENTATION DEFINED address space. It contains the following section:

- *About the System MMU implementation defined address space on page 12-164*.

12.1 About the System MMU IMPLEMENTATION DEFINED address space

The System MMU IMPLEMENTATION DEFINED address space is reserved for IMPLEMENTATION DEFINED purposes.

The size of this address space is defined by *PAGESIZE*. See *PAGESIZE and NUMPAGENDXB on page 9-98*.

The content of this address space is relative to the offset from address SMMU_GID_BASE, as Table 12-1 shows.

Table 12-1 System MMU IMPLEMENTATION DEFINED address space

Offset	Name	Description
`0x00000`-(*PAGESIZE* − `0x4`)	IMPLEMENTATION DEFINED	Reserved for IMPLEMENTATION DEFINED purposes

Chapter 13
System MMU Performance Monitors Extension Register Map

This chapter describes the recommended memory-mapped and external debug interface to the Performance Monitors Extension. It contains the following sections:

13.1 SMMU Performance Monitors Extension register summary

A memory-mapped interface is available for software running on any processor in the system to access counters in the Performance Monitors Extension. Table 13-1 shows a list of the memory-mapped registers for the Performance Monitors Extension.

Table 13-1 System MMU Performance Monitors Extension register map

Offset	Name	Type	Description
0x00000+*m*	PMEVCNTRn	RW	*PMEVCNTRn, Performance Monitors Event Count Registers on page 13-177*. *m* is 4 times the event counter number.
(0x00000+*m*)-0x003FC	-	-	Reserved. *m* is 4 times the number of event counters.
0x00400+*m*	PMEVTYPERn	RW	*PMEVTYPERn, Performance Monitors Event Type Registers on page 13-177*. *m* is 4 times the event counter number.
(0x00400+*m*)-0x007FC	-	-	Reserved. *m* is 4 times the number of event counters.
0x00800+*m*	PMCGCRn	RW[a]	*PMCGCRn, Performance Monitors Counter Group Configuration Registers on page 13-173*. *m* is 4 times the Counter group number.
(0x00800+*m*)-0x009FC	-	-	Reserved. *m* is 4 times the number of Counter groups.
0x00A00+*m*	PMCGSMRn	RW	*PMCGSMRn, Performance Monitors Counter Group Stream Match Registers on page 13-174*. *m* is 4 times the Counter group number.
(0x00A00+*m*) -0x00BFC	-	-	Reserved. *m* is 4 times the number of Counter groups.
0x00C00-0x00C1C	PMCNTENSETx	RW	*PMCNTENSETx, Performance Monitors Count Enable Set registers on page 13-175*.
0x00C20-0x00C3C	PMCNTENCLRx	RW	*PMCNTENCLRx, Performance Monitors Counter Enable Clear registers on page 13-174*.
0x00C40-0x00C5C	PMINTENSETx	RW	*PMINTENSETx, Performance Monitors Interrupt Enable Set registers on page 13-179*.
0x00C60-0x00C7C	PMINTENCLRx	RW	*PMINTENCLRx, Performance Monitors Interrupt Enable Clear registers on page 13-178*.
0x00C80-0x00C9C	PMOVSCLRx	RW	*PMOVSCLRx, Performance Monitors Overflow Status Clear registers on page 13-180*.
0x00CA0-0x00CBC	-	-	Reserved.
0x00CC0-0x00CDC	PMOVSSETx	RW	*PMOVSSETx, Performance Monitors Overflow Status Set registers on page 13-180*.
0x00CE0-0x00D7C	-	-	Reserved.
0x00D80-0x00DFC	-	-	IMPLEMENTATION DEFINED.
0x00E00	PMCFGR	RO[a]	*PMCFGR, Performance Monitors Configuration Register on page 13-172*.
0x00E04	PMCR	RW	*PMCR, Performance Monitors Control Register on page 13-176*.
0x00E08	-	-	Reserved.
0x00E0C-0x00E1C	-	-	Reserved.

Table 13-1 System MMU Performance Monitors Extension register map (continued)

Offset	Name	Type	Description
0x00E20	PMCEID0	RO	*PMCEID0, Performance Monitors Common Event Identifier 0 register on page 13-171*, *PMCEID1, Performance Monitors Common Event Identifier 1 register on page 13-172*.
0x00E24	PMCEID1		
0x00E28-0x00E7C	-	-	Reserved.
0x00E80-0x00EFC	IMPLEMENTATION DEFINED	-	-
0x00F00	-	-	Reserved for integration mode control register.
0x00F04-0x00FB4	-	-	Reserved.
0x00FB8	PMAUTHSTATUS	RO	*PMAUTHSTATUS, Performance Monitors Authentication Status register on page 13-169*.
0x00FBC-0x00FC8	-	-	Reserved.
0x00FCC	PMDEVTYPE	RO	*PMDEVTYPE, Performance Monitors Device Type Register on page 13-177*.
0x00FD0-0x00FFC	PMPIDy, PMCIDz	RO	Performance Monitors Peripheral Identification and Component Identification Registers.
0x01000-(*PAGESIZE*–0x4)	Reserved	-	-

a. See individual field descriptions for variations.

13.2 SMMU Performance Monitors Extension register descriptions

This section describes the System MMU Performance Monitors Extension registers that might be present in a System MMU implementation. Registers are shown in register name order.

The following applies to some of the registers described in this section:

- PMCFGR.N indicates the number of event counters, where:
  - counters are numbered continuously from 0 to PMCFGR.N
  - the number of implemented instances of the specified register is (PMCFGR.N DIV 32) +1.
- For performance monitor counter *i*, use:
  - register *x*, where *x* = *i* DIV 32
  - register bit *j*, where *j* = *i* MOD 32.

    For register *x*, *j* takes values from 0 to (*m*-1), where *m* is defined by the following pseudocode:

```
if (x == PMCFGR.N DIV 32) then
    m = (PMCFGR.N MOD 32) + 1;
else if (x ≥ PMCFGR.N DIV 32) then
    m = 0;  // the register is RAZ/WI
else
    x = 32; // all bits are RW.
```

The affected registers are:

- PMCNTENSETx
- PMCNTENCLRx
- PMINTENSETx
- PMINTENCLRx
- PMOVSCLRx
- PMOVSSETx.

13.2.1 PMAUTHSTATUS, Performance Monitors Authentication Status register

The PMAUTHSTATUS characteristics are:

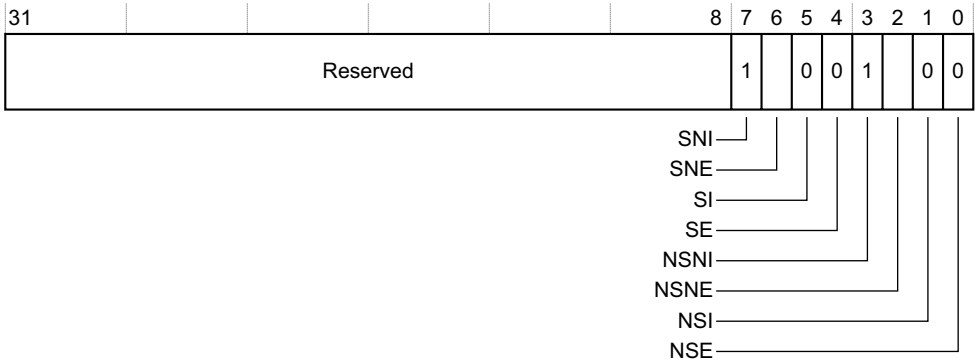
Purpose Indicates the implemented debug features and provides the current values of the configuration inputs that determine the debug permissions.

Usage constraints There are no usage constraints.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RO register.

In an implementation that includes the Security Extensions, the PMAUTHSTATUS bit assignments are:



Bits[31:8] Reserved.

SNI, bit[7] Secure non-invasive debug features implemented. This bit is RAO, Secure non-invasive debug features are implemented.

SNE, bit[6] Secure non-invasive debug enabled. This bit indicates whether counting of Secure transactions is permitted. For the recommended external debug interface, this bit is the logical result of (**DBGEN** OR **NIDEN**) AND (**SPIDEN** OR **SPNIDEN**).

SI, bit[5] Secure invasive debug features implemented. This bit is RAZ, Secure invasive debug features are not implemented.

SE, bit[4] Secure invasive debug enabled. This bit is RAZ.

NSNI, bit[3] Non-secure non-invasive debug features implemented. This bit is RAO, Non-secure non-invasive debug features are implemented.

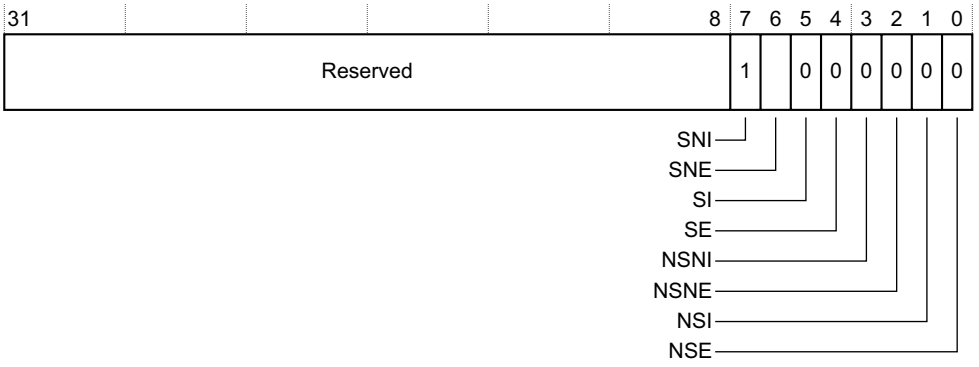
NSNE, bit[2] Non-secure non-invasive debug enabled. For the recommended external debug interface, this bit indicates the logical result of **DBGEN** OR **NIDEN**.

NSI, bit[1] Non-secure invasive debug features implemented. This bit is RAZ, Non-secure invasive debug features are not implemented.

NSE, bit[0] Non-secure invasive debug enabled. This bit is RAZ.

See *Interaction with the Security Extensions on page 7-88* for more information.

In an implementation that does not include the Security Extensions, the PMAUTHSTATUS bit assignments are:



Bits[31:8] Reserved.

SNI, bit[7] Secure non-invasive debug features implemented. This bit is RAO, Secure non-invasive debug features are implemented.

SNE, bit[6] Secure non-invasive debug enabled. This bit indicates whether non-invasive debug is permitted in Secure PL1 modes. For the recommended external debug interface, this bit is the logical result of **DBGEN** OR **NIDEN**.

SI, bit[5] Secure invasive debug features implemented. This bit is RAZ, Secure invasive debug features are not implemented.

SE, bit[4] Secure invasive debug enabled. This bit is RAZ. It indicates whether invasive halting debug is permitted in Secure PL1 modes.

NSNI, bit[3] Non-secure non-invasive debug features implemented. This bit is RAZ, Non-secure non-invasive debug features are not implemented.

NSNE, bit[2] Non-secure non-invasive debug enabled. This bit is RAZ, Non-secure non-invasive debug is not enabled.

NSI, bit[1] Non-secure invasive debug features implemented. This bit is RAZ, Non-secure invasive debug features are not implemented.

NSE, bit[0] Non-secure invasive debug enabled. This bit is RAZ, Non-secure invasive debug is not enabled.

13.2.2 PMCEID0, Performance Monitors Common Event Identifier 0 register

The PMCEID0 register characteristics are:

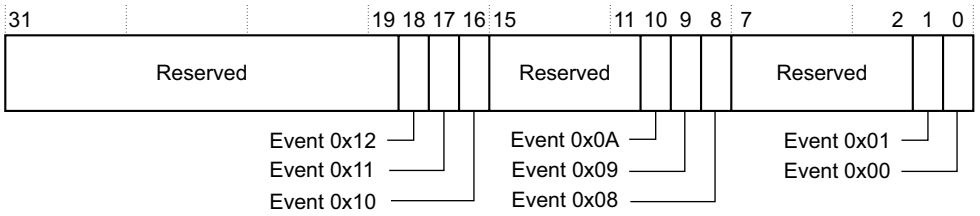
Purpose Describes the event classes supported by the System MMU implementation.

Usage constraints There are no usage constraints.

Configurations See *About the System MMU Performance Monitors Extension on page 7-80* for information about when this register is configured.

Attributes A 32-bit RO register.

The PMCEID0 bit assignments are:



Bits[31:19] Reserved.

Event 0x12, bit[18]

The possible values of this bit are:

0 Access write event not implemented.

1 Access write event implemented.

Event 0x11, bit[17]

The possible values of this bit are:

0 Access read event not implemented.

1 Access read event implemented.

Event 0x10, bit[16]

The possible values of this bit are:

0 Access event not implemented.

1 Access event implemented.

Bits[15:11] Reserved.

Event 0x0A, bit[10]

The possible values of this bit are:

0 TLB refill write event not implemented.

1 TLB refill write event implemented.

Event 0x09, bit[9]

The possible values of this bit are:

0 TLB refill read event not implemented.

1 TLB refill read event implemented.

Event 0x08, bit[8]

The possible values of this bit are:

0 TLB refill event not implemented.

1 TLB refill event implemented.

Bits[7:2] Reserved.

Event 0x01, bit[1]

The possible values of this bit are:

0 Cycle count divided by 64 event not implemented.

1 Cycle count divided by 64 event implemented.

Event 0x00, bit[0]

The possible values of this bit are:

0 Cycle count event not implemented.

1 Cycle count event implemented.

13.2.3 PMCEID1, Performance Monitors Common Event Identifier 1 register

Whether a System MMU supports the Performance Monitors Extension is IMPLEMENTATION DEFINED:

- If this extension is supported, all of the bits in this 32-bit read-only register are reserved and are UNK.
- If this extension is not supported, the register map corresponding to the Performance Monitors registers is UNK/SBZP.

13.2.4 PMCFGR, Performance Monitors Configuration Register

The PMCFGR characteristics are:

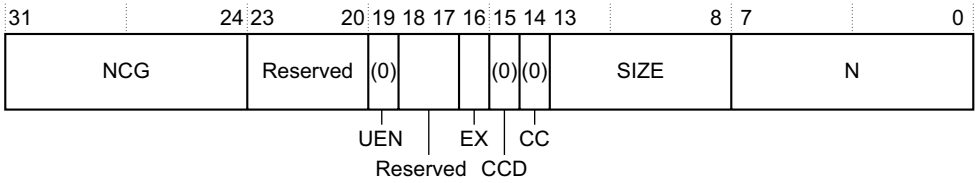
Purpose Provides *Performance Monitoring Unit* (PMU) configuration data.

Usage constraints There are no usage constraints.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RO register.

The PMCFGR bit assignments are:



NCG, bit[31:24] Number of Counter Groups.

The number of Counter groups implemented is calculated as NCG+1. For example, if NCG==`0x00`, there is one implemented Counter group.

Bits[23:20] Reserved.

UEN, bit[19] Unprivileged-mode Enable, reads as the value 0 indicating that this feature is not supported.

Bits[18:17] Reserved.

EX, bit[16] Event export. The possible values of this bit are:

0 Event export is not supported. PMCR.X is RAZ/WI.

1 Event export is supported. PMCR.X is writable.

CCD, bit[15] Cycle Counter pre-scale, reads as the value 0 indicating that there is no cycle counter pre-scale.

CC, bit[14] Cycle Counter, reads as the value 0 indicating that a dedicated cycle counter is not implemented.

SIZE, bits[13:8]

Counter size, reads as the value `0b011111` indicating 32-bit event counters.

N, bits[7:0] Indicates the number of implemented event counters, up to a maximum of 256 counters. The number of counters implemented is N+1.

Note

In the CPU PMU, the number of counters implemented is N. This is because the CPU PMU implements an extra counter, PMCCNTR, which is not defined in the System MMU architecture.

13.2.5 PMCGCR*n*, Performance Monitors Counter Group Configuration Registers

The PMCGCR*n* characteristics are:

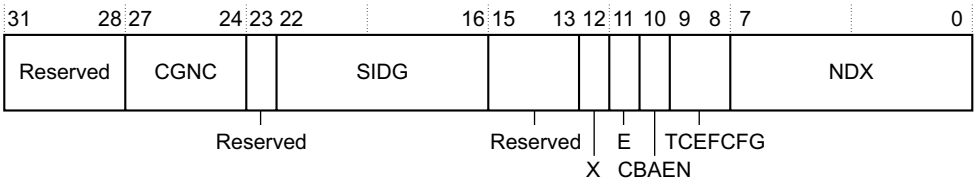
Purpose Controls Counter group behavior.

Usage constraints There are no usage constraints.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The PMCGCR*n* bit assignments are:



Bits[31:28] Reserved.

CGNC, bits[27:24]

Counter Group Number of Counters, is the number of counters in this Counter group, in the range 1-15, so that a value of `0b0001` == 1 counter.

This field is RO/WI.

Bit[23] Reserved.

SIDG, bits[22:16]

StreamID Group, indicates the StreamID group that this Counter group is affiliated with.

This field is RO/WI.

Bits[15:13] Reserved.

X, bit[12] Export, corresponds to the Performance Monitors Event Export field, SMMU_CBn_PMCR.X, for this Counter group.

E, bit[11] Count Enable, corresponds to the Performance Monitors Count Enable field, SMMU_CBn_PMCR.E, for this Counter group.

CBAEN, bit[10]

Context Bank Assignment Enable.

The possible values of this bit are:

0 Do not reveal Counter group *n* in the Translation context bank specified by PMCGCR*n*.NDX.

1 Reveal Counter group *n* in the Translation context bank specified by PMCGCR*n*.NDX.

If PMCGCR*n*.CBAEN==1 and PMCGCR*n*.TCEFCFG != (`0b10` or `0b01`), behavior is UNPREDICTABLE.

TCEFCFG, bits[9:8]

Translation Context Event Filtering Configuration.

The possible values of this bit are:

`0b00` Count events on a global basis.

`0b01` Counter events are restricted to matches in the corresponding PMCGSMRn register.

`0b10` Counter events are restricted to the Translation context bank indicated by PMCGCR*n*.NDX.

`0b11` Reserved.

NDX, bits[7:0] Index, only relevant when PMCGCRn.CBAEN==0b1 or PMCGCRn.TCEFCFG==0b10. Otherwise, PMCGCR*n*.NDX is SBZ.

13.2.6 PMCGSMR*n*, Performance Monitors Counter Group Stream Match Registers

The PMCGSMR*n* characteristics are:

Purpose Specifies StreamID filtering of the events counted in a Counter group.

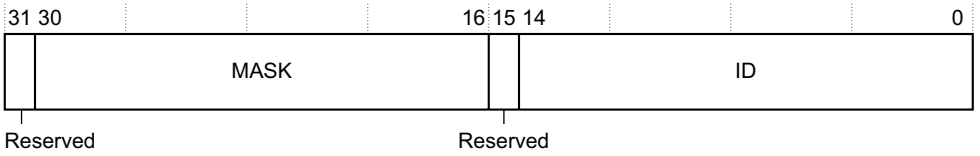
For information about enabling StreamID event filtering, see the corresponding PMCGCRn.TCEFCFG field.

Usage constraints There are no usage constraints.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The PMCGSMR*n* bit assignments are:



Bit[31] Reserved.

MASK, bits[30:16]

if MASK[i] == 1, ID[i] is ignored and SBZ.

if MASK[i] == 0, ID[i] is valid for matching.

The number of MASK bits actually present is IMPLEMENTATION DEFINED. Unimplemented bits are RAZ/WI.

Bit[15] Reserved.

ID, bits[14:0] The StreamID to match.

The number of ID bits actually present is IMPLEMENTATION DEFINED. Unimplemented bits are RAZ/WI.

13.2.7 PMCNTENCLR*x*, Performance Monitors Counter Enable Clear registers

The PMCNTENCLR*x* registers characteristics are:

Purpose Disables any implemented event counter, PMN*i*, for *i* in the range ($x \times 32$) to ($x \times 32$) + 31.

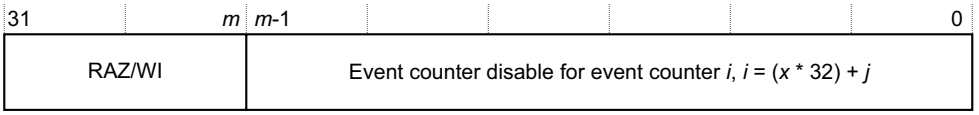
Reading this register shows which counters are enabled.

Usage constraints Used in conjunction with PMCNTENSETx.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The PMCNTENCLR*x* register bit assignments are:



Note

See *SMMU Performance Monitors Extension register descriptions on page 13-168* for the definitions of *m*, *x* and *j*.

Bits[31:*m*] RAZ/WI.

P*i*, bit[*j*], for *j* = 0 to (*m*–1)

Event counter *i* disable bit, where $i = (x \times 32) + j$.

Table B4-3 shows the behavior of this bit on reads and writes.

Table 13-2 Read and write values for the PMCNTENCLR*x*.P*i* bits

P*x* value	Meaning on read	Action on write
0	PMN*i* event counter disabled.	No action, write is ignored.
1	PMN*i* event counter enabled.	Disable the PMN*i* event counter.

Note

PMCR.E can override the settings in this register and disable all counters. PMCNTENCLR*x* retains its value when PMCR.E is 0, even though its settings are ignored.

13.2.8 PMCNTENSET*x*, Performance Monitors Count Enable Set registers

The PMCNTENSET*x* registers characteristics are:

Purpose Enables any implemented event counters, PMN*i*, for *i* in the range $(x \times 32)$ to $(x \times 32) + 31$.

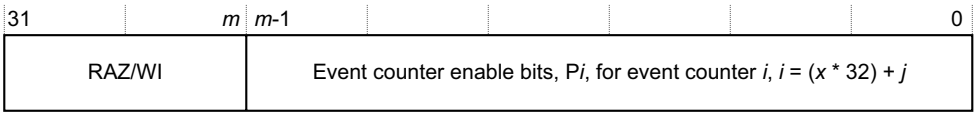
Reading a PMCNTENSET*x* register shows which counters are enabled.

Usage constraints Use in conjunction with PMCNTENCLRx.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The PMCNTENSET*x* register bit assignments are:



Note

See *SMMU Performance Monitors Extension register descriptions on page 13-168* for the definitions of *m*, *x* and *j*.

Bits[31:*m*] RAZ/WI.

P*i*, bit[*j*], for *j* = 0 to (*m*–1)

Event counter *i* enable bit, where $i = (x \times 32) + j$.

Table B4-5 shows the behavior of this bit on reads and writes.

Table 13-3 Read and write values for the PMCNTENSET*x*.P*i* bits

P*x* value	Meaning on read	Action on write
0	PMN*i* event counter disabled.	No action, write is ignored.
1	PMN*i* event counter enabled.	Enable the PMN*i* event counter.

13.2.9 PMCR, Performance Monitors Control Register

The PMCR characteristics are:

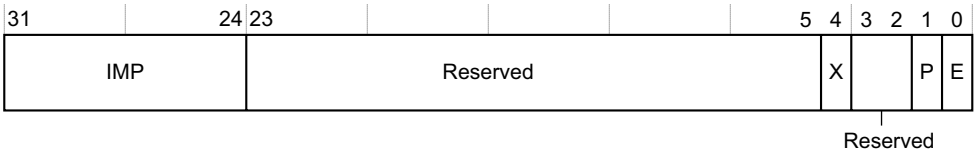
Purpose PMCR provides controls for the Performance Monitors.

Usage constraints There are no usage constraints.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The PMCR bit assignments are:



IMP, bits[31:24] Implementor code that is either:
- allocated by ARM
- the value 0.

This RO field is IMPLEMENTATION DEFINED.

Bits[23:5] Reserved.

X, bit[4] Export enable. The possible values of this bit are:

0 Export of events is disabled.

1 Export of events is enabled.

See *SMMU_CBn_PMCR banking on page 7-86* for information about PMCR banking.

Bits[3:2] Reserved.

P, bit[1] Event counter reset. This bit is WO. The effects of writing to this bit are:

0 No action.

1 Reset all event counters to zero.

See *SMMU_CBn_PMCR banking on page 7-86* for information about PMCR banking.

E, bit[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.

See *SMMU_CBn_PMCR banking on page 7-86* for information about PMCR banking.

13.2.10 PMDEVTYPE, Performance Monitors Device Type Register

The PMDEVTYPE characteristics are:

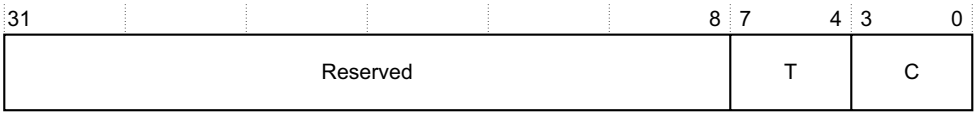
Purpose Provides the CoreSight device type information for the Performance Monitors, and indicates the type of debug component.

Usage constraints There are no usage constraints.

Configurations Must be implemented in all CoreSight components.

Attributes A 32-bit RO register.

The PMDEVTYPE bit assignments are:



Bits[31:0] Reserved.

T, bits[7:4] Sub-type, a fixed value of `0x5`, which indicates association with a memory management unit conforming to the ARM System MMU Architecture.

C, bits[3:0] Class, a fixed value of `0x6`, which indicates a Performance Monitor device type.

13.2.11 PMEVCNTR*n*, Performance Monitors Event Count Registers

The PMEVCNTR*n* characteristics are:

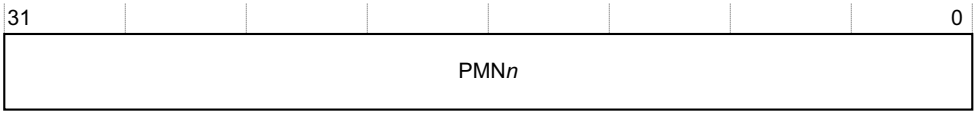
Purpose Reads or writes the value of the selected event counter, PMN*n*.

Usage constraints There are no usage constraints.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The PMEVCNTR*n* bit assignments are:



PMNx, bits[31:0] The value of the selected event counter, PMN*n*.

Note

PMEVCNTR*n* can be written to by software even when the counter is disabled. This is true regardless of why the counter is disabled, which can be any of:

- because 1 has been written to the appropriate bit in PMCNTENCLRx
- because PMCR.E is 0.

13.2.12 PMEVTYPER*n*, Performance Monitors Event Type Registers

The PMEVTYPER*n* characteristics are:

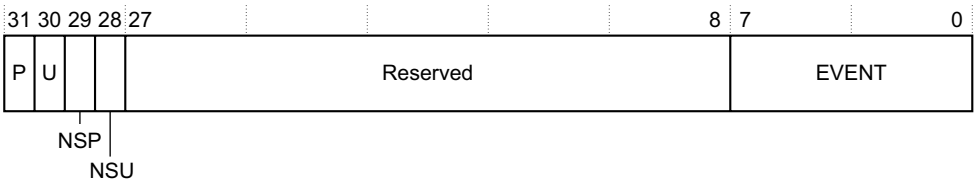
Purpose Controls which events are counted by the corresponding event counter.

Usage constraints There are no usage constraints.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The PMEVTYPER*n* bit assignments are:



P, bit[31] Privileged transactions filtering bit. Controls the counting of Secure privileged transactions.

The possible values of this bit are:

0 Count events relating to Secure privileged transactions.

1 Do not count events relating to Secure privileged transactions.

U, bit[30] Unprivileged transactions filtering bit. Controls the counting of Secure unprivileged transactions.

The possible values of this bit are:

0 Count events relating to Secure unprivileged transactions.

1 Do not count events relating to Secure unprivileged transactions.

NSP, bit[29] Non-secure Privileged transactions filtering bit. Controls the counting of Non-secure privileged transactions.

The possible values of this bit are:

P==NSP Count events relating to Non-secure privileged transactions.

P!=NSP Do not count events relating to Non-secure privileged transactions.

NSU, bit[28] Non-secure unprivileged transactions filtering bit. Controls counting of Non-secure unprivileged transactions.

The possible values of this bit are:

U==NSU Count events relating to Non-secure unprivileged transactions.

U!=NSU Do not count events relating to Non-secure unprivileged transactions.

Bits[27:8] Reserved.

EVENT, bits[7:0]

Event type. See *Event classes on page 7-82*.

13.2.13 PMINTENCLR*x*, Performance Monitors Interrupt Enable Clear registers

The PMINTENCLR*x* registers characteristics are:

Purpose Disables the generation of interrupt requests on overflows from each implemented event counter, PMN*i*, for *i* in the range ($x \times 32$) to ($x \times 32$) + 31.

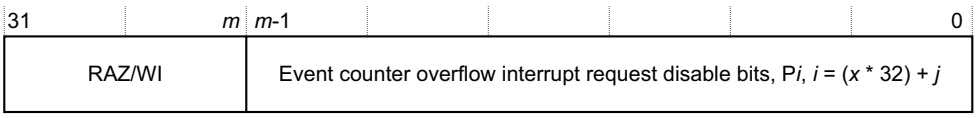
Reading the register shows which overflow interrupt requests are enabled.

Usage constraints Used in conjunction with the PMINTENSETx register.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The PMINTENCLR*x* register bit assignments are:



Note

See *SMMU Performance Monitors Extension register descriptions on page 13-168* for the definitions of *m*, *x* and *j*.

Bits[31:*m*] RAZ/WI.

P*i*, bit[*j*], for *j* = 0 to (*m*–1)

Event counter *i*, overflow interrupt request disable bit, where $i = (x \times 32) + j$.

Table B4-7 shows the behavior of this bit on reads and writes.

Table 13-4 Read and write values for the PMINTENCLR*x*.P*i* bits

P*i* value	Meaning on read	Action on write
0	PMN*i* interrupt request disabled.	No action, write is ignored.
1	PMN*i* interrupt request enabled.	Disable the PMN*i* interrupt request.

13.2.14 PMINTENSET*x*, Performance Monitors Interrupt Enable Set registers

The PMINTENSET*x* registers characteristics are:

Purpose Enables the generation of interrupt requests on overflows from each implemented event counter, PMN*i*, for *i* in the range $(x \times 32)$ to $(x \times 32) + 31$.

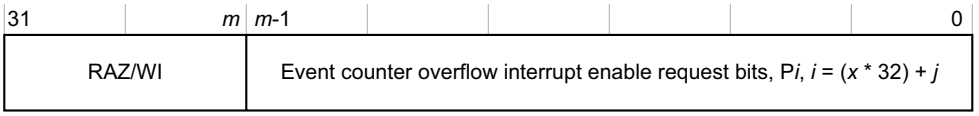
Reading PMINTENSET*x* shows which overflow interrupt requests are enabled.

Usage constraints Used in conjunction with PMINTENCLRx.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The PMINTENSET*x* register bit assignments are:



Note

See *SMMU Performance Monitors Extension register descriptions on page 13-168* for the definitions of *m*, *x* and *j*.

Bits[31:*m*] RAZ/WI.

P*i*, bit[*j*], for *j* = 0 to (*m*–1)

Event counter *i*, overflow interrupt request enable bit, where $i = (x \times 32) + j$.

Table B4-9 shows the behavior of this bit on reads and writes.

Table 13-5 Read and write values for the PMINTENSET*x*.P*i* bits

P*i* value	Meaning on read	Action on write
0	PMN*i* interrupt request disabled.	No action, write is ignored.
1	PMN*i* interrupt request enabled.	Enable the PMN*i* interrupt request.

When an interrupt is signaled, software can remove it by writing a 1 to the corresponding overflow bit in PMOVSCLRx.

Note

ARM expects that the interrupt request that can be generated on a counter overflow is exported from the processor, meaning it can be factored into a system interrupt controller if applicable. This means that normally the system has more levels of control of the interrupt generated.

13.2.15 PMOVSCLR*x*, Performance Monitors Overflow Status Clear registers

The PMOVSCLR*x* registers characteristics are:

Purpose Clears the state of the overflow bit for each implemented event counter, PMN*i*, for *i* in the range ($x \times 32$) to ($x \times 32$) + 31.

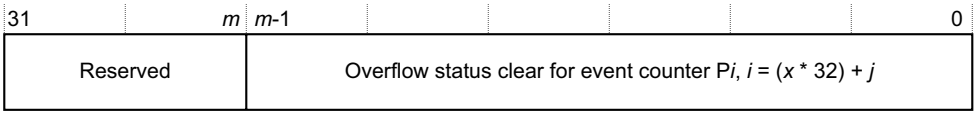
Reading the register shows the current overflow status.

Usage constraints There are no usage constraints.

Configurations Implemented only as part of the Performance Monitors Extension.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The PMOVSCLR*x* bit assignments are:



Note

See *SMMU Performance Monitors Extension register descriptions on page 13-168* for the definitions of *m*, *x* and *j*.

Bits[31:*m*] RAZ/WI.

P*i*, bit[*j*], for *j* = 0 to (*m*–1)

Event counter *i*, overflow status clear bit, where $i = (x \times 32) + j$.

Table B4-7 shows the behavior of this bit on reads and writes.

Table 13-6 Read and write values for the PMOVSCLR*x*.P*i* bits

P*i* value	Meaning on read	Action on write
0	Counter PMN*i* has not overflowed.	No action, write is ignored.
1	Counter PMN*i* has overflowed.	Set bit to 0.

13.2.16 PMOVSSET*x*, Performance Monitors Overflow Status Set registers

The PMOVSSET*x* registers characteristics are:

Purpose Sets the state of the overflow bit for each of the implemented event counters, PMN*i*, for *i* in the range ($x \times 32$) to ($x \times 32$) + 31.

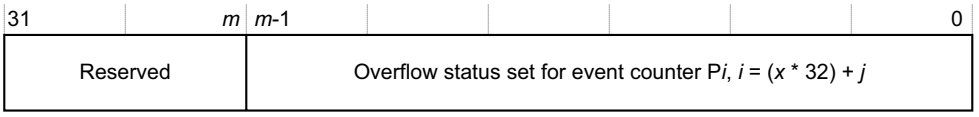
Reading the register shows the current overflow status.

Usage constraints Implemented only as part of the Performance Monitors Extension.

Configurations See *About the System MMU Performance Monitors Extension on page 7-80* for information about when this register is configured.

Attributes A 32-bit RW register with an UNKNOWN reset value.

The PMOVSET*x* bit assignments are:



Note

See *SMMU Performance Monitors Extension register descriptions on page 13-168* for the definitions of *m*, *x* and *j*.

Bits[31:*m*] RAZ/WI.

P*i*, bit[*j*], for *j* = 0 to (*m*–1)

Event counter *i*, overflow status set bit, where $i = (x \times 32) + j$.

Table B4-7 shows the behavior of this bit on reads and writes.

Table 13-7 Read and write values for the PMINTENCLR*x*.P*i* bits

P*i* value	Meaning on read	Action on write
0	Counter PMN*i* has not overflowed.	No action, write is ignored.
1	Counter PMN*i* has overflowed.	Set bit to 1.

13.2.17 PMPID*y*, PMCID*z*, Performance Monitors Peripheral ID and Component ID registers

The Peripheral ID and Component ID registers, PMPID0-PMPID7 and PMCID0-PMCID3, are RO registers that provide standard CoreSight peripheral and component identification. For details, see the definitions of DBGPID*n* and DBGCID*n* in the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition*.

Note

The part number defined by PMPID1[3:0] and PMPID0[7:0] must be a different part number to that defined by DBGPID1[3:0] and DBGPID0[7:0].

Chapter 14
The Security State Determination Address Space

This chapter gives an overview of the security state determination address space. It contains the following section:

- *System MMU security state determination address space on page 14-184*.

14.1 System MMU security state determination address space

The security state determination address space can indicate whether each SSD_Index is acting for the Secure domain or the Non-secure domain. This address space is only accessible to Secure memory transactions.

One bit is provided for each SSD_Index value. A System MMU implementation supports an SSD_Index of up to 15 bits in size. This corresponds 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. Security state determination register bits corresponding to values above the implemented SSD_Index size behave as UNK/WI.

Security state determination bits can have fixed values that correspond to SSD_Index values having 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 is IMPLEMENTATION DEFINED. A System MMU implementation and the system it is integrated in can use alternative means to resolve the Secure status of transactions. SMMU_IDR1.SSDTP indicates the presence of the security state determination table. In an implementation where the registers are not present, this address space behaves as UNK/WI.

The security state determination address space layout is shown in terms of offsets from *SMMU_GSSD_BASE*. See Table 14-1.

Table 14-1 Security state determination address space

Offset	Name	Description
0x00000	SMMU_SSDR0	Corresponds to SSD_Index values 0 - 31
0x00004	SMMU_SSDR1	Corresponds to SSD_Index values 32 - 63
0x00008 - 0x00FFC	SMMU_SSDR1023 - SMMU_SSDR2	Corresponds to SSD_Index values 64 - 32767
0x01000 - (*PAGESIZE* − 0x4)	Reserved	-

If the security state determination register space is implemented, the behavior of each SMMU_SSDR*n* 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
}
```

Chapter 15
Stage 1 Translation Context Bank Format

This chapter specifies the format of a Stage 1 Translation context bank. It contains the following sections:

15.1 Stage 1 Translation context bank format summary

The Stage 1 Translation context bank format is defined in terms of offsets from the Translation context bank base address, *SMMU_CBn_BASE*, as Table 15-1 shows.

Table 15-1 Stage 1 Translation context bank format

Offset	Name	Type	Description
0x00000	SMMU_CBn_SCTLR	RW	*SMMU_CBn_SCTLR, System Control Register on page 15-211*
0x00004	SMMU_CBn_ACTLR	RW	*SMMU_CBn_ACTLR, Auxiliary Control Register on page 15-194*
0x00008	SMMU_CBn_RESUME	WO	Resume or terminate a stalled transaction, *SMMU_CBn_RESUME, Transaction Resume register on page 15-210*
0x0000C-0x0001C	Reserved	-	-
0x00020	SMMU_CB*n*_TTBR0[31:0]	RW[a]	*SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224*
0x00024	SMMU_CB*n*_TTBR0[63:32]	RW[a]	64-bit *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224*
0x00028	SMMU_CB*n*_TTBR1[31:0]	RW[a]	*SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224*
0x0002C	SMMU_CB*n*_TTBR1[63:32]	RW[a]	64-bit, *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224*
0x00030	SMMU_CBn_TTBCR	RW	*SMMU_CBn_TTBCR, Translation Table Base Control Register on page 15-221*
0x00034	SMMU_CBn_CONTEXTIDR	RW	*SMMU_CBn_CONTEXTIDR, Context Identification Register on page 15-196*
0x00038	SMMU_CBn_PRRR or SMMU_CBn_MAIR0	RW	*SMMU_CBn_PRRR, Primary Region Remap Register on page 15-208*, *SMMU_CBn_MAIRm, Memory Attribute Indirection Registers on page 15-201*
0x0003C	SMMU_CBn_NMRR or SMMU_CBn_MAIR1	RW	*SMMU_CBn_NMRR, Normal Memory Remap Register on page 15-201*, *SMMU_CBn_MAIRm, Memory Attribute Indirection Registers on page 15-201*
0x00040-0x00044	Reserved		Reserved for IMPLEMENTATION DEFINED memory attributes associated with memory encodings. Such attributes are only additional qualifiers to the memory locations, and cannot change the architected behaviors of the memory attributes defined in the SMMU_CB*n*_MAIR0 and SMMU_CB*n*_MAIR1 registers. See *SMMU_CBn_MAIRm, Memory Attribute Indirection Registers on page 15-201* for more information.
0x00048-0x0004C	Reserved	-	-
0x00050	SMMU_CBn_PAR[31:0]	RW	*SMMU_CBn_PAR, Physical Address Register on page 15-202*
0x00054	SMMU_CBn_PAR[63:32]		
0x00058	SMMU_CBn_FSR	RW	*SMMU_CBn_FSR, Fault Status Register on page 15-197*
0x0005C	SMMU_CBn_FSRRESTORE	WO	*SMMU_CBn_FSRRESTORE, Fault Status Restore Register on page 15-198*

Table 15-1 Stage 1 Translation context bank format (continued)

Offset	Name	Type	Description
0x00060	SMMU_CBn_FAR[31:0]	RW	*SMMU_CBn_FAR, Fault Address Register on page 15-197*
0x00064	SMMU_CBn_FAR[63:32]		
0x00068	SMMU_CB*n*_FSYNR0	RW	See *SMMU_CBn_FSYNRm, Fault Syndrome Registers on page 15-199*
0x0006C	SMMU_CB*n*_FSYNR1		
0x00070-0x005FC	Reserved	-	-
0x00600	SMMU_CBn_TLBIVA[31:0]	WO	*SMMU_CBn_TLBIVA, Invalidate TLB by VA on page 15-216*
0x00604	Reserved	-	-
0x00608	SMMU_CBn_TLBIVAA[31:0]	WO	*SMMU_CBn_TLBIVAA, TLB Invalidate by VA All ASID on page 15-217*
0x0060C	Reserved	-	-
0x00610	SMMU_CBn_TLBIASID	WO	*SMMU_CBn_TLBIASID, TLB Invalidate by ASID on page 15-216*
0x00614	Reserved	-	-
0x00618	SMMU_CBn_TLBIALL	WO	*SMMU_CBn_TLBIALL, TLB Invalidate All on page 15-215*
0x0061C	Reserved	-	-
0x00620	SMMU_CBn_TLBIVAL[31:0]	WO	*SMMU_CBn_TLBIVAL, TLB Invalidate by VA, Last level on page 15-218*
0x00624	Reserved	-	-
0x00628	SMMU_CBn_TLBIVAAL[31:0]	WO	*SMMU_CBn_TLBIVAAL, TLB Invalidate by VA, All ASID, Last level on page 15-218*
0x0062C-0x007EC	Reserved	-	-
0x007F0	SMMU_CBn_TLBSYNC	WO	*SMMU_CBn_TLBSYNC, TLB Synchronize Invalidate on page 15-220*
0x007F4	SMMU_CBn_TLBSTATUS	RO	*SMMU_CBn_TLBSTATUS, TLB Status on page 15-219*
0x007F8-0x007FC	Reserved	-	-
0x00800	SMMU_CBn_ATS1PR[31:0]	WO	*SMMU_CBn_ATS1PR, Address Translation Stage 1 Privileged Read on page 15-194*
0x00804	Reserved	-	-
0x00808	SMMU_CBn_ATS1PW[31:0]	WO	*SMMU_CBn_ATS1PW, Address Translation Stage 1 Privileged Write on page 15-194*
0x0080C	Reserved	-	-
0x00810	SMMU_CBn_ATS1UR[31:0]	WO	*SMMU_CBn_ATS1UR, Address Translation Stage 1 Unprivileged Read on page 15-195*
0x00814	Reserved	-	-
0x00818	SMMU_CBn_ATS1UW[31:0]	WO	*SMMU_CBn_ATS1UW, Address Translation Stage 1 Unprivileged Write on page 15-195*

Table 15-1 Stage 1 Translation context bank format (continued)

Offset	Name	Type	Description
0x0081C-0x008EC	Reserved	-	-
0x008F0	SMMU_CBn_ATSR	RO	*SMMU_CBn_ATSR, Address Translation Status Register on page 15-196*
0x008F4-0x00CFC	Reserved	-	-
0x00D00-0x00DFC	IMPLEMENTATION DEFINED	RW	Reserved for IMPLEMENTATION DEFINED purposes
0x00E00-0x00E38	SMMU_CBn_PMEVCNTRm	RW	*SMMU_CBn_PMEVCNTRm, Performance Monitors Event Counter registers on page 15-207*
0x00E3C-0x00E7C	Reserved	-	-
0x00E80-0x00EB8	SMMU_CBn_PMEVTYPERm	RW	*SMMU_CBn_PMEVTYPERm, Performance Monitors Event Type Registers on page 15-207*
0x00EBC-0x00EFC	Reserved	-	-
0x00F00	SMMU_CBn_PMCFGR	RO	*SMMU_CBn_PMCFGR, Performance Monitors Configuration Register on page 15-206*
0x00F04	SMMU_CBn_PMCR	RW	*SMMU_CBn_PMCR, Performance Monitors Control Register on page 15-207*
0x00F08-0x00F1C	Reserved	-	-
0x00F20	SMMU_CB*n*_PMCEID0	RO	*SMMU_CBn_PMCEIDm, Performance Monitors Common Event Identification registers on page 15-206*
0x00F24	SMMU_CB*n*_PMCEID1	RO	*SMMU_CBn_PMCEIDm, Performance Monitors Common Event Identification registers on page 15-206*
0x00F28-0x00F3C	Reserved	-	-
0x00F40	SMMU_CBn_PMCNTENSET	RW	*SMMU_CBn_PMCNTENSET, Performance Monitors Count Enable Set register on page 15-207*
0x00F44	SMMU_CBn_PMCNTENCLR	RW	*SMMU_CBn_PMCNTENCLR, Performance Monitors Count Enable Clear register on page 15-206*
0x00F48	SMMU_CBn_PMINTENSET	RW	*SMMU_CBn_PMINTENSET, Performance Monitors Interrupt Enable Set register on page 15-208*
0x00F4C	SMMU_CBn_PMINTENCLR	RW	*SMMU_CBn_PMINTENCLR, Performance Monitors Interrupt Enable Clear register on page 15-208*
0x00F50	SMMU_CBn_PMOVSCLR	RW	*SMMU_CBn_PMOVSCLR, Performance Monitors Overflow Status Clear Register on page 15-208*
0x00F54	Reserved	-	-
0x00F58	SMMU_CBn_PMOVSSET	-	*SMMU_CBn_PMOVSSET, Performance Monitors Overflow Status Set Register on page 15-208*
0x00F5C-0x00FB4	Reserved	-	-
0x00FB8	SMMU_CBn_PMAUTHSTATUS	RO	*SMMU_CBn_PMAUTHSTATUS, Performance Monitors Authentication Status register on page 15-206*

Table 15-1 Stage 1 Translation context bank format (continued)

Offset	Name	Type	Description
0x00FBC-0x00FCC	Reserved	-	-
0x00FD0-0x00FFC	IMPLEMENTATION DEFINED	-	Reserved for Primecell ID
0x01000 - (*PAGESIZE* − 0x4)	Reserved	-	-

a. Mainly RW. See field descriptions for detail.

15.2 Reset values

In each System MMU Stage 1 Translation context bank, the value of each register field is UNPREDICTABLE after a system reset, with the exceptions shown in Table 15-2.

Table 15-2 Predictable post-reset values

Field	Reset value	Notes
SMMU_CBn_FSR.SS	0	No requirement to perform a retry or termination operation.
SMMU_CBn_SCTLR.CFIE	0	Prevent interrupt assertion as a result of UNPREDICTABLE error status.
SMMU_CBn_SCTLR.CFRE	0	Prevent abort as a result of UNPREDICTABLE error status.

15.3 Memory attribute indirection

The SMMU_CBn_MAIRm memory indirection attribute system provides a revised version of the *Type EXtension* (TEX) Remap system to redirect the selection of memory attributes from the translation table entries. This system takes the 3-bit MemAttr field in the translation table entry, and uses it to access an 8-bit field in one of the 32-bit Memory Attribute Indirection Registers, SMMU_CB*n*_MAIR0 and SMMU_CB*n*_MAIR1. See the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition* for more information about the TEX Remap system.

SMMU_CB*n*_MAIR0 is selected when AttrIndex[2] is set to the value 0.

SMMU_CB*n*_MAIR1 is selected when AttrIndex[2] is set to 1.

The field selected in SMMU_CB*n*_MAIR0 and SMMU_CB*n*_MAIR1 is determined by AttrIndex[1:0], as Bits[8×AttrIndx[1:0] + 7:8×AttrIndx[1:0]].

The encoding of each MAIR[7:4] field is shown in Table 15-3, where:

- *R* denotes the Outer Read-Allocate policy.
- *W* denotes the Outer Write-Allocate policy.

The allocation policy is encoded so that a value of 0 means do not allocate, and a value of 1 means allocate.

Table 15-3 MAIR encodings

Bits[7:4]	Meaning
`0b0000`	Strongly-ordered or Device memory.
`0b00`*RW*	Outer Write-Through transient Normal memory. *RW*=`0b00` is not supported.
`0b0100`	Outer Non-cacheable Normal memory.
`0b01`*RW*	Outer Write-Back transient Normal memory. *RW*=`0b00` is not supported.
`0b10`*RW*	Outer Write-Through non-transient Normal memory.
`0b11`*RW*	Outer Write-Back non-transient Normal memory.

The transient variants provide a hint that the data being accessed has a transient access property. The memory system might adjust its handling of the data accordingly.

All other values are UNPREDICTABLE.

Table 15-4 shows the encoding of each MAIR[3:0] field, where:

- *R* denotes the Inner Read-Allocate policy
- *W* denotes the Inner Write-Allocate policy.

The allocation policy is encoded so that a value of 0 means do not allocate, and a value of 1 means allocate.

Table 15-4 MAIR encodings

Bits[3:0]	Meaning when bits[7:4] == `0b0000`	Meaning when bits[7:4] != `0b0000`
`0b0000`	Strongly-ordered	UNPREDICTABLE.
`0b00`*RW*	UNPREDICTABLE	Inner Write-Through transient Normal memory. *RW*=`0b00` is not supported.
`0b0100`	Device	Inner Non-cacheable Normal memory.

Table 15-4 MAIR encodings (continued)

Bits[3:0]	Meaning when bits[7:4] == 0b0000	Meaning when bits[7:4] != 0b0000
0b01*RW*	UNPREDICTABLE	Inner Write-Back transient Normal memory. *RW*=0b00 is not supported.
0b10*RW*	UNPREDICTABLE	Inner Write-Through non-transient Normal memory.
0b11*RW*	UNPREDICTABLE	Inner write-back non-transient Normal memory.

All other values are UNPREDICTABLE.

15.4 Multi-format registers and reserved fields

Some registers have a number of possible formats, where the format depends on the format that the register has been configured to, and the format of the data recorded in the register. See *Multi-format registers and reserved fields on page 10-114* for more information, with the exception that in a Stage 1 Translation context bank, the register format is affected by SMMU_CBn_TTBCR.EAE instead of by a TYPE field.

15.5 Stage 1 Translation context bank register descriptions

This section describes all of the Stage 1 Translation context bank registers that might be present in a System MMU implementation. Registers are shown in register name order.

15.5.1 SMMU_CB*n*_ACTLR, Auxiliary Control Register

The SMMU_CB*n*_ACTLR characteristics are:

Purpose Provides implementation-specific configuration and control options.

Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit RW register with an UNPREDICTABLE reset value.

The SMMU_CB*n*_ACTLR bit assignments are IMPLEMENTATION DEFINED.

15.5.2 SMMU_CB*n*_ATS1PR, Address Translation Stage 1 Privileged Read

The SMMU_CB*n*_ATS1PR characteristics are:

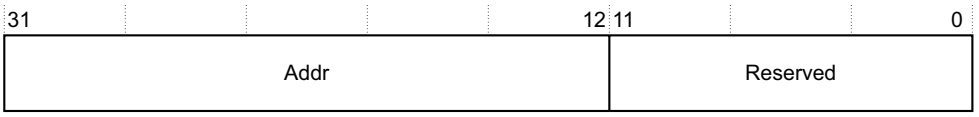
Purpose Translates an argument-supplied input address and writes the result to SMMU_CBn_PAR.

Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit WO command. Reads are SBZ.

The SMMU_CB*n*_ATS1PR bit assignments are:



Addr, bits[31:12]

Addr, the input address.

Bits[11:0] Reserved.

15.5.3 SMMU_CB*n*_ATS1PW, Address Translation Stage 1 Privileged Write

The SMMU_CB*n*_ATS1PW characteristics are:

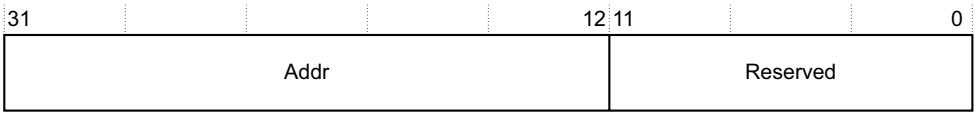
Purpose Translates the argument-supplied input address and writes the result to SMMU_CBn_PAR.

Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit WO command. Reads are SBZ.

The SMMU_CB*n*_ATS1PW bit assignments are:



Addr, bits[31:12]

Addr, the input address.

Bits[11:0] Reserved.

15.5.4 SMMU_CBn_ATS1UR, Address Translation Stage 1 Unprivileged Read

The SMMU_CB*n*_ATS1UR characteristics are:

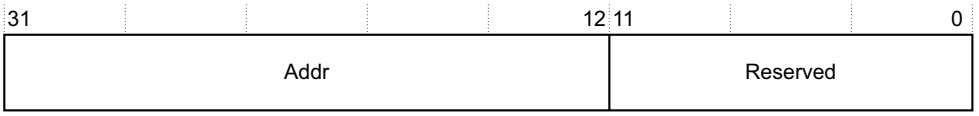
Purpose Translates the argument-supplied input address and writes the result to SMMU_CBn_PAR.

Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit WO command. Reads are SBZ.

The SMMU_CB*n*_ATS1UR bit assignments are:



Addr, bits[31:12]

Addr, the input address.

Bits[11:0] Reserved.

15.5.5 SMMU_CBn_ATS1UW, Address Translation Stage 1 Unprivileged Write

The SMMU_CB*n*_ATS1UW characteristics are:

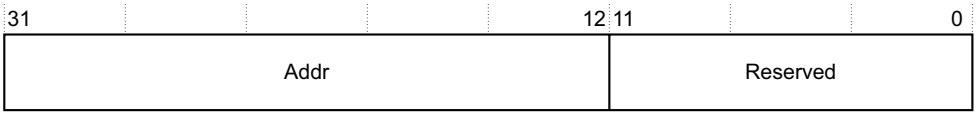
Purpose Translates the argument-supplied input address and writes the result to SMMU_CBn_PAR.

Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit WO command. Reads are SBZ.

The SMMU_CB*n*_ATS1UW bit assignments are:



Addr, bits[31:12] Addr, the input address.

Bits[11:0] Reserved.

15.5.6 SMMU_CBn_ATSR, Address Translation Status Register

The SMMU_CB*n*_ATSR characteristics are:

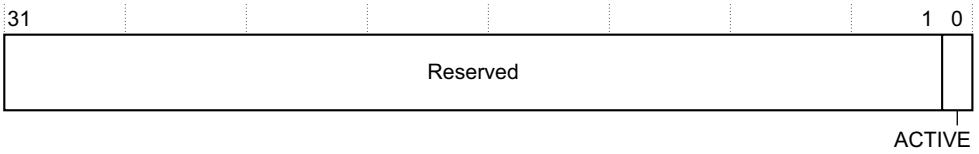
Purpose Provides status information about active address translation operations for a Translation context bank.

Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit RO register with an UNPREDICTABLE reset value. Writes are ignored.

The SMMU_CB*n*_ATSR bit assignments are:



Bits[31:1] Reserved.

ACTIVE, bit[0]

Address Translation Active. The possible values of this bit are:

0 Operation not active.

1 Operation active. SMMU_CBn_PAR contents are yet to be updated.

15.5.7 SMMU_CB*n*_CONTEXTIDR, Context Identification Register

The SMMU_CB*n*_CONTEXTIDR characteristics are:

Purpose Identifies the current process identifier and the current address space identifier.

Usage constraints No usage constraints apply.

Configurations When the Short-descriptor translation table format is selected, that is, SMMU_CBn_TTBCR.EAE has the value 0, the ASID is used by many memory management functions. The PROCID field has no hardware usage within the System MMU.

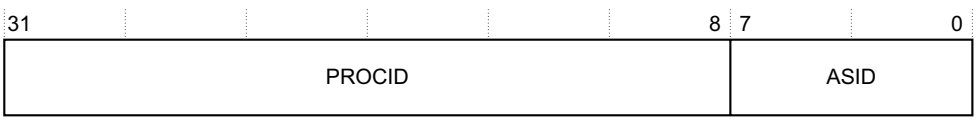
When the Long-descriptor translation table format is selected, that is, SMMU_CBn_TTBCR.EAE has the value 1, the ASID field is not present in this register, and the MMU refers to the SMMU_CBn_TTBRm.ASID fields instead.

It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit RW register with an UNPREDICTABLE reset value.

Register format when SMMU_CBn_TTBCR.EAE is 0

The SMMU_CB*n*_CONTEXTIDR bit assignments in this format are:



PROCID, bits[31:8]

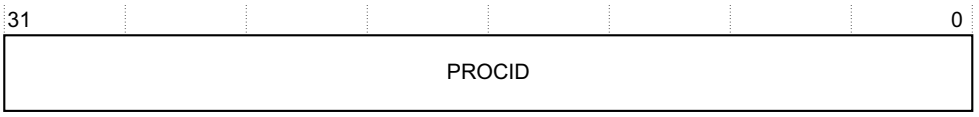
Process Identifier.

ASID, bits[7:0]

Address Space Identifier.

Register format when SMMU_CBn_TTBCR.EAE is 1

The SMMU_CB*n*_CONTEXTIDR bit assignments in this format are:



PROCID, bits[31:0]

Process Identifier.

15.5.8 SMMU_CB*n*_FAR, Fault Address Register

The SMMU_CB*n*_FAR characteristics are:

Purpose Holds the input address of the memory access that caused a synchronous abort exception.

Usage constraints No usage constraints apply.

Configurations For a Stage 1 Translation context bank, the value of *N* is 32, equating to 4GB.

For a Stage 2 Translation context bank, the value of *N* depends on the size of the input address implemented by the System MMU, and can be:

- 32, for 4GBytes
- 36, for 64GBytes
- 40, for 1TByte.

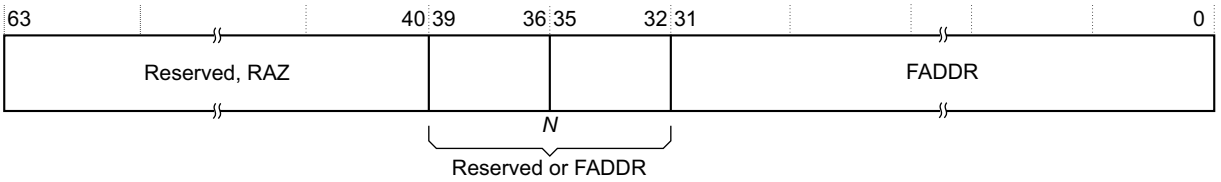
Unimplemented bits behave as RAZ/WI.

SMMU_IDR2.IAS specifies the implemented input address space.

It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit RW register with an UNPREDICTABLE reset value.

The SMMU_CB*n*_FAR bit assignments are:



Bits[31:*N*] Reserved.

FADDR, bits[*N*-1:0]

Fault Address, the input address of the faulting access.

15.5.9 SMMU_CB*n*_FSR, Fault Status Register

The SMMU_CB*n*_FSR characteristics are:

Purpose Provides memory system fault status information.

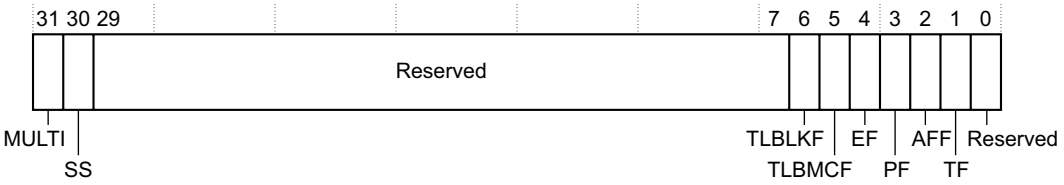
Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit RW clear register. A value of 1 written to any non-reserved bit clears that bit. A value of 0 written to any of these bits leaves the bit unchanged.

With the exception of SMMU_CB*n*_FSR[30], the fields in this register have UNPREDICTABLE reset values.

The SMMU_CB*n*_FSR bit assignments are:



MULTI, bit[31]

Multiple Faults, indicates that an additional context fault occurred while the value in SMMU_CB*n*_FSR was nonzero.

SS, bit[30] Stalled Status. The possible values of this read-only bit are:

0 The context is not stalled.

1 The context is stalled because of an exception in the context bank.

When the context is stalled, this bit is set to 1. A write to SMMU_CBn_RESUME is the only way to reset this bit, and depending on the value written, the write either resumes or terminates the stalled transaction.

Note

This bit is not affected by any SMMU_CBn_FSRRESTORE operation.

This bit resets to 0.

Bits[29:7] Reserved.

TLBLKF, bit[6]

TLB Lock Fault. The possible values of this bit are:

0 There is no TLB lock fault.

1 A TLB lock fault has occurred.

TLBMCF, bit[5]

TLB Match Conflict Fault. The possible values of this bit are:

0 There is no TLB Match conflict fault.

1 A fault caused by multiple matches was detected in the TLB.

EF, bit[4] External Fault. The possible values of this bit are:

0 There is no External fault.

1 An External fault has occurred.

PF, bit[3] Permission Fault. The possible values of this bit are:

0 There is no Permission fault.

1 A fault caused by insufficient permission to complete a memory access has occurred.

AFF, bit[2] Access Flag Fault. The possible values of this bit are:

0 There is no Access flag fault.

1 A fault caused by the access flag being set for the address being accessed has occurred.

TF, bit[1] Translation Fault. The possible values of this bit are:

0 There is no Translation fault.

1 A Translation fault has occurred. The mapping for the address being accessed is invalid.

Bit[0] Reserved.

15.5.10 SMMU_CB*n*_FSRRESTORE, Fault Status Restore Register

The SMMU_CB*n*_FSRRESTORE characteristics are:

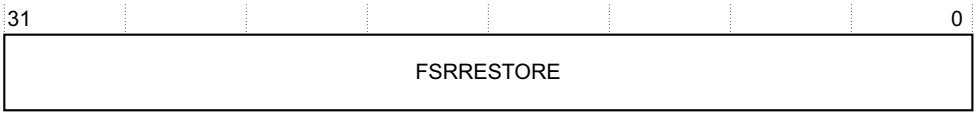
Purpose Restores the contents of SMMU_CBn_FSR.

Usage constraints The bit corresponding to the SMMU_CBn_FSR.SS bit is ignored. The only way to reset the SS bit is by writing to SMMU_CBn_RESUME.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit WO register.

The SMMU_CB*n*_FSRRESTORE bit assignments are:



FSRRESTORE, bits[31:0] Data written to FSRRESTORE is shown in SMMU_CBn_FSR.

15.5.11 SMMU_CBn_FSYNR*m*, Fault Syndrome Registers

The SMMU_CB*n*_FSYNR1 and SMMU_CB*n*_FSYNR0 characteristics are:

Purpose Holds fault syndrome information about the memory access that caused a synchronous abort exception.

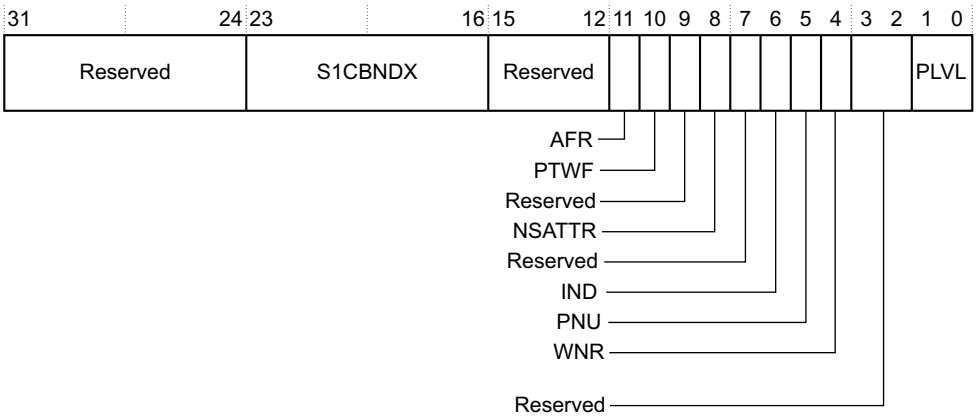
Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes 32-bit RW registers with UNPREDICTABLE reset values.

SMMU_CB*n*_FSYNR0

The SMMU_CB*n*_FSYNR0 bit assignments are:



Bits[31:24] Reserved.

S1CBNDX[23:16]

Stage 1 Context Bank Index associated with the transaction that caused the fault.

For nested translation, this field contains the Stage 1 Translation context bank index for processing the transaction.

For stage 2 only translation, this field is UNKNOWN.

This field is only present in a stage 2 format Translation context bank. In a stage 1 format Translation context bank, it is UNK/SBPZ.

This field is only valid if SMMU_IDR0.NTS==1. In an implementation that does not include nested translation, this field is UNK/WI.

Bits[15:12] Reserved.

AFR, bit[11] Asynchronous Fault Recorded. The possible values of this bit are:

0 A fault was recorded synchronously.

1 A fault was recorded asynchronously.

PTWF, bit[10] A walk fault on a translation table access. The possible values of this bit are:

0 A walk fault did not occur.

1 A fault occurred during processing of a translation table walk.

Bit[9] Reserved.

NSATTR, bit[8]

Non-secure Attribute. The possible values of this bit are:

0 The input transaction has a Secure attribute.

1 The input transaction has a Non-secure attribute.

In a Non-secure context bank, this bit is reserved. In a Secure context bank, this bit records the attributes of the transaction after applying the SMMU_S2CRn.NSCFG bit transformation. See *Recording memory attributes on page 3-47* for more information.

Bit[7] Reserved.

IND, bit[6] Instruction Not Data. The possible values of this bit are:

0 Data.

1 Instruction.

This bit records the attributes of the transaction after applying the SMMU_S2CRn.INSTCFG bit transformation. See *Recording memory attributes on page 3-47* for more information.

PNU, bit[5] Privileged Not Unprivileged. The possible values of this bit are:

0 Unprivileged.

1 Privileged.

This bit records the attributes of the transaction after applying the SMMU_S2CRn.PRIVCFG bit transformation. See *Recording memory attributes on page 3-47* for more information.

WNR, bit[4] Write Not Read. The possible values of this bit are:

0 Read.

1 Write.

Bits[3,2] Reserved.

PLVL, bits[1:0]

Translation Table Level, the level in the translation table walk that the fault is associated with. The encoding of this field is:

`0b01` Level 1.

`0b10` Level 2.

`0b11` Level 3.

Faults and translation table level association

The translation table level a fault is associated with is:

- For a fault associated with a translation table walk, the level of table walk being performed.
- For a translation fault, the level of translation table that gave the fault. If a disabled translation table walk causes the fault or if the size of the address presented is out of the range specified for matching with any base address register, the fault is reported for level 1.
- For an access fault, the level of translation table that gave the fault.

- For a permission fault, including a fault caused by a hierarchical permission, the final level of translation table used for that translation.

SMMU_CB*n*_FSYNR1

The 32-bit SMMU_CB*n*_FSYNR1 register bit assignments are IMPLEMENTATION DEFINED.

15.5.12 SMMU_CB*n*_MAIR*m*, Memory Attribute Indirection Registers

The SMMU_CB*n*_MAIR1 and SMMU_CB*n*_MAIR0 characteristics are:

Purpose Provide a revised version of the TEX-Remap system to redirect the selection of memory attributes from the translation table entries.

Usage constraints No usage constraints apply.

Configurations These registers are included when the Long-descriptor format translation tables are selected, that is, when SMMU_CBn_TTBCR.EAE has the value 1.

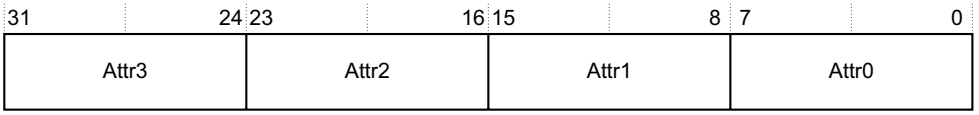
When the legacy Short-descriptor translation table format is selected, SMMU_CB*n*_MAIR1 and SMMU_CB*n*_MAIR0 are replaced by SMMU_CBn_PRRR and SMMU_CBn_NMRR.

See *Memory attribute indirection on page 15-191*.

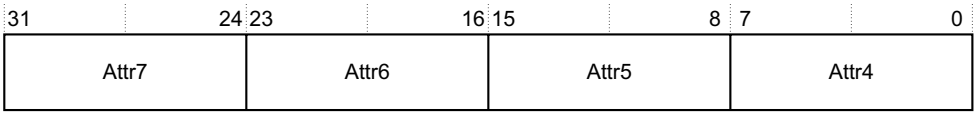
It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes 32-bit RW registers with UNPREDICTABLE reset values.

The SMMU_CB*n*_MAIR0 bit assignments are:



The SMMU_CB*n*_MAIR1 bit assignments are:



15.5.13 SMMU_CB*n*_NMRR, Normal Memory Remap Register

The SMMU_CB*n*_NMRR characteristics are:

Purpose Provides additional mapping controls for memory regions that are mapped as Normal memory by their entry in SMMU_CBn_PRRR.

Usage constraints No usage constraints apply.

Configurations SMMU_CB*n*_NMRR is included when the legacy Short-descriptor translation table format is selected. That is, when SMMU_CBn_TTBCR.EAE is set to the value 0. When the Long-descriptor translation table format is selected, SMMU_CB*n*_NMRR is replaced by SMMU_CB*n*_MAIR1. See *SMMU_CBn_MAIRm, Memory Attribute Indirection Registers*.

It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit RW register with an UNPREDICTABLE reset value.

The SMMU_CB*n*_NMRR bit assignments are:

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
OR7	OR6	OR5	OR4	OR3	OR2	OR1	OR0	IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0

OR7 - OR0, bits[31:16]

Outer cacheable property mapping for the memory attributes in *m*, if the region is mapped as Normal memory by the TR*m* entry in SMMU_CBn_PRRR. *m* is the value of the TEX[0], C and B bits in the translation table.

The encoding of this field is:

`0b00` Non-cacheable.

`0b01` Write-back, Write-Allocate.

`0b10` Write-Through, no Write-Allocate.

`0b11` Write-back, no Write-Allocate.

IR7 - IR0, bits[15:0]

Inner cacheable property mapping for the memory attributes in *m*, if the region is mapped as Normal memory by the TR*m* entry in SMMU_CBn_PRRR. *m* is the value of the TEX[0], C and B bits in the translation table. The possible values of this field are the same as for those of the SMMU_CB*n*_NMRR.OR*m* fields. The meaning of the field with *m* = 6 is IMPLEMENTATION DEFINED, and might differ from the meaning given in this specification. This is because the meaning of the attribute combination TEX[0] = 1, C = 1, B = 0 is IMPLEMENTATION DEFINED.

15.5.14 SMMU_CB*n*_PAR, Physical Address Register

The SMMU_CB*n*_PAR characteristics are:

Purpose Receives the physical address during any virtual to physical address translation.

Usage constraints No usage constraints apply.

Configurations The SMMU_CB*n*_PAR format depends on whether the Short-descriptor or Long-descriptor translation table format is selected, and whether the transaction completes successfully. See:

- *Short-descriptor translation table format on page 15-203*
- *Long-descriptor translation table format on page 15-204*
- *Fault format on page 15-205*.

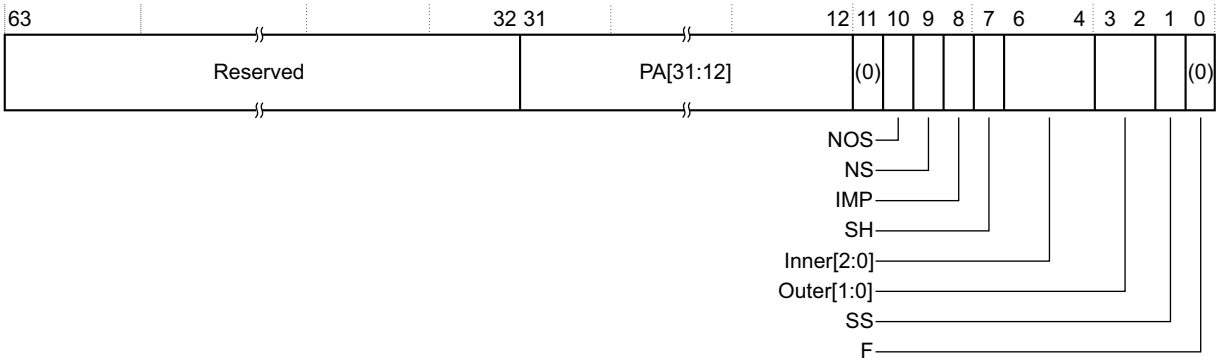
See also *Multi-format registers and reserved fields on page 15-193*.

It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 64-bit RW register with an UNPREDICTABLE reset value.

Short-descriptor translation table format

If the translation completes successfully, the bit assignments are:



Bits[63:32] Reserved.

PA[31:12], bits[31:12]

Bits[31:12] of the physical address.

Bit[11], 0 Reserved with the value 0.

NOS, bit[10] Not Outer Shareable attribute, indicates whether the physical memory is Outer Shareable. The possible values of this bit are:

0 Memory is Outer Shareable.

1 Memory is not Outer Shareable.

Whether an implementation distinguishes between Inner Shareable and Outer Shareable memory is IMPLEMENTATION DEFINED. If an implementation does not make this distinction, this field is UNK/SBZP.

NS, bit[9] Non-Secure, the NS attribute for a translation table entry read from a Secure Translation context bank.

This bit is UNKNOWN for a translation table entry read from a Non-secure Translation context bank.

IMP, bit[8] This bit is IMPLEMENTATION DEFINED.

SH, bit[7] Shareable attribute, indicates whether the physical memory is shareable.

The possible values of this bit are:

0 Memory is not shareable

1 Memory is shareable.

Inner[2:0], bits[6:4]

Inner memory attributes from the translation table entry.

The encoding of this field is:

`0b111` Write-back, no Write-Allocate.

`0b110` Write-Through.

`0b101` Write-back, Write-Allocate.

`0b011` Device.

`0b001` Strongly-ordered.

`0b000` Non-cacheable.

All other encodings for Inner[2:0] are reserved.

Outer[1:0], bits[3:2]

Outer memory attributes from the translation table entry.

The encoding of this field is:

`0b00` Write-back, no Write-Allocate.

`0b01` Write-Through, no Write-Allocate.

`0b10` Write-back, Write-Allocate.

`0b11` Non-cacheable.

SS, bit[1] Super Section, indicates if the result is a supersection.

The possible values of this bit are:

0 The page is not a supersection. PAR[31:12] contains PA[31:12], regardless of the page size.

1 The page is part of a supersection, where:

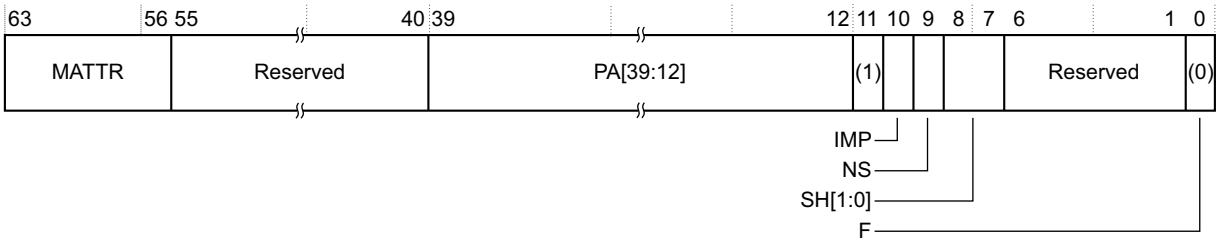
- PAR[31:24] contains PA[31:24].
- PAR[23:16] contains PA[39:32].
- PAR[15:12] contains `0b0000`.

PA[23;12] is the same as VA[23:12] for supersections.

F(0), bit[0] Fault. This bit is 0 if the translation completes successfully.

Long-descriptor translation table format

If the translation completes successfully, the bit assignments are:



MATTR, bits[63:56]

Memory Attributes. See *SMMU_CBn_MAIRm, Memory Attribute Indirection Registers on page 15-201* for more information.

Bits[55:40] Reserved.

PA[39:12], bits[39:12]

Bits 39:12 of the physical address.

Bit[11], 1 Reserved with the value 1.

IMP, bit[10] IMPLEMENTATION DEFINED.

NS, bit[9] Non-Secure, the NS attribute for a translation table entry read from a Secure Translation context bank.

This bit is UNKNOWN for a translation table entry read from a Non-secure Translation context bank.

SH[1:0], bits[8:7]

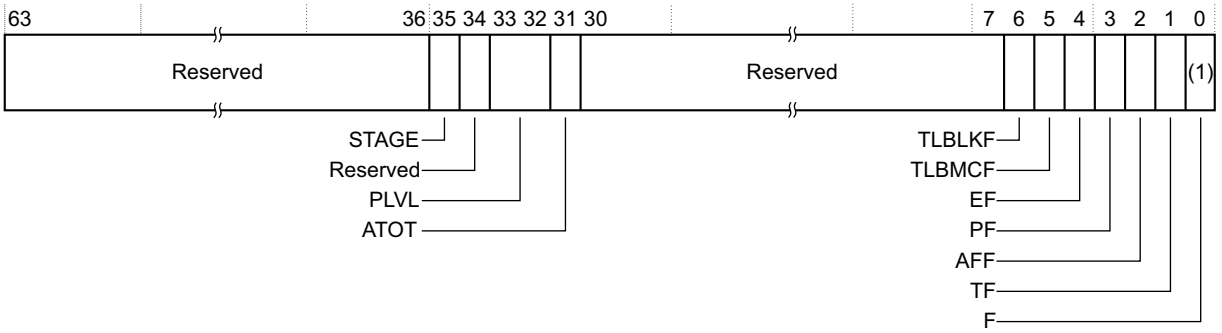
Shareability.

Bits[6:1] Reserved.

F(0) Fault. This bit is 0 if the translation completes successfully.

Fault format

If the translation fails to complete successfully, the bit assignments are, irrespective of translation format:



Bits[63:36] Reserved.

STAGE, bit[35]

Stage of translation that encountered the fault. The possible values of this bit are:

0 Stage 1.

1 Stage 2.

Bit[34] Reserved.

PLVL, bits[33:32]

Page level, the level of translation table walk that encountered the fault. The encoding of this field is:

`0b01` Level 1.

`0b10` Level 2.

`0b11` Level 3.

ATOT, bit[31] Address Translation Operation Terminated. The possible values of this bit are:

0 This fault does not apply.

1 The address translation operation was terminated. The requested operation could not be completed.

Bits[30:7] Reserved.

TLBLKF, bit[6]

TLB Lock Fault. The possible values of this bit are:

0 This fault does not apply.

1 TLB lock fault applies.

TLBMCF, bit[5]

TLB Match Conflict Fault. The possible values of this bit are:

0 This fault does not apply.

1 Fault caused by multiple matches detected in the TLB.

EF, bit[4] External Fault. The possible values of this bit are:

0 This fault does not apply.

1 Fault caused by an external fault on a translation table walk.

PF, bit[3] Permission Fault. The possible values of this bit are:

0 This fault does not apply.

1 Fault caused by insufficient permission to complete the memory access.

AFF, bit[2] Access Flag Fault. The possible values of this bit are:

0 This fault does not apply.

1 Access flag fault occurred.

TF, bit[1] Translation Fault. The possible values of this bit are:

0 This does not apply.

1 Translation fault condition applies. Invalid mapping for address being accessed.

F(1), bit[0] Fault. This bit is set to 1 if the translation aborts.

15.5.15 SMMU_CB*n*_PMAUTHSTATUS, Performance Monitors Authentication Status register

The Performance Monitors Authentication Status Register, SMMU_CB*n*_PMAUTHSTATUS, provides the equivalent of the PMAUTHSTATUS register, in the register map of a Translation context bank.

If a Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMAUTHSTATUS has the same format as PMAUTHSTATUS.

If no Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMAUTHSTATUS is UNK/SBZP.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.16 SMMU_CB*n*_PMCEID*m*, Performance Monitors Common Event Identification registers

The 32-bit RO Performance Monitors Common Event Identification registers, SMMU_CB*n*_PMCEID0 and SMMU_CB*n*_PMCEID1, provide the equivalent of the System MMU performance monitoring register map PMCEID0 and PMCEID1 registers, in the register map of a Translation context bank.

If a Counter group is revealed in a Translation context bank, SMMU_CB*n*_PMCEID0 and SMMU_CB*n*_PMCEID1 have the same format as PMCEID0 and PMCEID1.

If no Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMCEID0 and SMMU_CB*n*_PMCEID1 are UNK.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.17 SMMU_CB*n*_PMCFGR, Performance Monitors Configuration Register

The 32-bit RO Performance Monitors Configuration Register, SMMU_CB*n*_PMCFGR, provides a performance monitoring configuration register in the register map of a Translation context bank.

If a Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMCFGR has a similar format to PMCFGR, with the following differences:

- NCG is UNK
- N indicates the number of event counters in the Counter group.

If no Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMCFGR is RAZ/WI. Software can examine SMMU_CB*n*_PMCFGR.SIZE to determine whether any counters are revealed:

- if (SMMU_CB*n*_PMCFGR.SIZE==`0b000000`) no counter group is revealed
- if (SMMU_CB*n*_PMCFGR.SIZE==`0b011111`) a counter group is revealed.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.18 SMMU_CBn_PMCNTENCLR, Performance Monitors Count Enable Clear register

The Performance Monitors Count Enable Clear register, SMMU_CB*n*_PMCNTENCLR, provides the equivalent of the PMCNTENCLRx register, in the register map of a Translation context bank.

If a Counter group is revealed in a Translation context bank:

- SMMU_CB*n*_PMCNTENCLR has the same format as PMCNTENCLRx
- bit *j* of SMMU_CB*n*_PMCNTENCLR controls event counter *j* in the corresponding Counter group.

If no Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMCNTENCLR is UNK/SBZP.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.19 SMMU_CB*n*_PMCNTENSET, Performance Monitors Count Enable Set register

The Performance Monitors Count Enable Set register, SMMU_CB*n*_PMCNTENSET, provides the equivalent of the PMCNTENSETx register, in the register map of a Translation context bank.

If a Counter group is revealed in a Translation context bank:

- SMMU_CB*n*_PMCNTENSET has the same format as PMCNTENSETx
- bit *j* of SMMU_CB*n*_PMCNTENSET controls event counter *j* in the corresponding Counter group.

If a no Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMCNTENSET is UNK/SBZP.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.20 SMMU_CB*n*_PMCR, Performance Monitors Control Register

The Performance Monitors Control Register, SMMU_CB*n*_PMCR, provides the equivalent of the PMCR register, in the register map of a Translation context bank.

If a Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMCR has the same format as PMCR.

If no Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMCR is UNK/SBZP.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.21 SMMU_CB*n*_PMEVCNTR*m*, Performance Monitors Event Counter registers

The Performance Monitors Event Counter Registers, SMMU_CB*n*_PMEVCNTR*m*, provide event counter resources in the register map of a Translation context bank.

If a Counter group is revealed in a Translation context bank:

- the event counter registers of the group are arranged in sequence starting from the first SMMU_CB*n*_PMEVCNTR*m* address offset
- the SMMU_CB*n*_PMEVCNTR*m* register format is as specified for PMEVCNTRn
- an SMMU_CB*n*_PMEVCNTR*m* register that corresponds to an offset greater than the number of event counter registers in the Counter group is UNK/SBZP.

If no Counter group is revealed in the Translation context bank, all of the SMMU_CB*n*_PMEVCNTR*m* registers are UNK/SBZP.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.22 SMMU_CB*n*_PMEVTYPER*m*, Performance Monitors Event Type Registers

The Performance Monitors Event Type Registers, SMMU_CB*n*_PMEVTYPER*m*, provide event type resources in the register map of a Translation context bank.

If a Counter group is revealed in the Translation context bank:

- the event type registers of that group are arranged in sequence, starting from the first SMMU_CB*n*_PMEVTYPER*m* address offset
- the SMMU_CB*n*_PMEVTYPER*m* register format is as specified for PMEVTYPERn
- an SMMU_CB*n*_PMEVTYPER*m* register that corresponds to an offset greater than the number of event type registers in the Counter Group is UNK/SBZP.

If no Counter group is revealed in the Translation context bank, all of the SMMU_CB*n*_PMEVTYPER*m* registers are WI.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.23 SMMU_CBn_PMINTENCLR, Performance Monitors Interrupt Enable Clear register

The Performance Monitors Interrupt Enable Clear Register, SMMU_CB*n*_PMINTENCLR, provides the equivalent of the PMINTENCLRx register, in the register map of a Translation context bank.

If a Counter group is revealed in the Translation context bank:

- SMMU_CB*n*_PMINTENCLR has the same format as PMINTENCLRx
- bit *j* of SMMU_CB*n*_PMINTENCLR corresponds to event counter *j* in the corresponding Counter group.

If no Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMINTENCLR is UNK/SBZP.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.24 SMMU_CB*n*_PMINTENSET, Performance Monitors Interrupt Enable Set register

The Performance Monitors Interrupt Enable Set register, SMMU_CB*n*_PMINTENSET, provides the equivalent of the PMINTENSETx register, in the register map of a Translation context bank.

If a Counter group is revealed in the Translation context bank:

- SMMU_CB*n*_PMINTENSET has the same format as PMINTENSETx
- bit *j* of SMMU_CB*n*_PMINTENSET corresponds to event counter *j* in the corresponding Counter group.

If no Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMINTENSET is UNK/SBZP.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.25 SMMU_CB*n*_PMOVSCLR, Performance Monitors Overflow Status Clear Register

The Performance Monitors Overflow Status Clear Register, SMMU_CB*n*_PMOVSCLR, provides the equivalent of the PMOVSCLRx register, in the register map of a Translation context bank.

If a Counter group is revealed in the Translation context bank:

- SMMU_CB*n*_PMOVSCLR has the same format as PMOVSCLRx
- bit *j* of SMMU_CB*n*_PMOVSCLR corresponds to event counter *j* in the corresponding Counter group.

If no Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMOVSCLR is UNK/SBZP.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.26 SMMU_CB*n*_PMOVSSET, Performance Monitors Overflow Status Set Register

The Performance Monitors Overflow Status Set Register, SMMU_CB*n*_PMOVSSET, provides the equivalent of PMOVSSETx, in the register map of a Translation context bank.

If a Counter group is revealed in the Translation context bank:

- SMMU_CB*n*_PMOVSSET has the same format as PMOVSSETx
- bit *j* of SMMU_CB*n*_PMOVSSET corresponds to event counter *j* in the corresponding Counter group.

If no Counter group is revealed in the Translation context bank, SMMU_CB*n*_PMOVSSET is UNK/SBZP.

For more information, see Chapter 7 *System MMU Performance Monitors Extension*.

15.5.27 SMMU_CB*n*_PRRR, Primary Region Remap Register

The SMMU_CB*n*_PRRR characteristics are:

Purpose Controls top-level mapping of the TEX[0], C, and B memory region attributes.

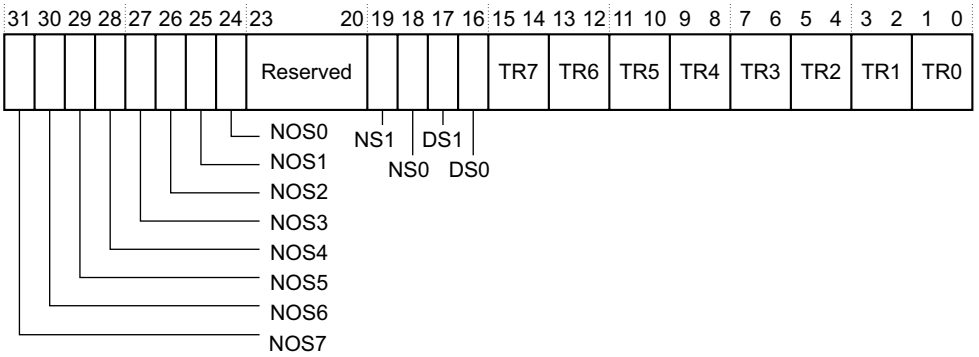
Usage constraints No usage constraints apply.

Configurations This register is included when the legacy Short-descriptor translation table format is selected. That is, when SMMU_CBn_TTBCR.EAE is 0. When the Long-descriptor translation table format is selected, is replaced by SMMU_CB*n*_MAIR0 replaces SMMU_CB*n*_PRRR. See *SMMU_CBn_MAIRm, Memory Attribute Indirection Registers on page 15-201*.

It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit RW register with an UNPREDICTABLE reset value.

The SMMU_CB*n*_PRRR bit assignments are:



NOS*m*, bits[31:24]

Outer Shareable property mapping for memory attributes in *n*, if the region is mapped as Normal memory that is Shareable where *n* is the value of the TEX[0] C and B bits in the translation table.

The possible value of each NOS bit is:

0 If the region is mapped as shareable Normal memory, the region is Outer Shareable.

1 The region is not Outer Shareable.

The meaning of the field where *m* = 6 is IMPLEMENTATION DEFINED and might differ from the meaning given in this specification. This is because the meaning of the attribute combination TEX[0] = 1, C = 1, B = 0 is IMPLEMENTATION DEFINED. If the implementation does not distinguish between Inner Shareable and Outer Shareable, these bits are reserved and are RAZ/WI.

Bits[23:20] Reserved.

NS1, bit[19] Mapping of the S = 1 attribute for Normal memory. This bit gives the mapped Shareable attribute for a region of memory that is mapped as Normal memory and has the S bit set to the value 1.

The possible values of this bit are:

0 The region is not shareable.

1 The region is shareable.

NS0, bit[18] Mapping of S = 0 attribute for Normal memory. This bit gives the mapped Shareable attribute for a region of memory that is mapped as Normal memory and has the S bit set to the value 0.

The possible values of this bit are:

0 The region is not shareable.

1 The region is shareable.

DS1, bit[17] Mapping of S = 1 attribute for Device memory. This bit gives the mapped Shareable attribute for a region of memory that is mapped as Device memory and has the S bit set to the value 0.

The possible values of this bit are:

0 The region is not shareable.

1 The region is shareable.

DS0, bit[16] Mapping of S = 0 attribute for Device memory. This bit gives the mapped Shareable attribute for a region of memory that is mapped as Device memory and has the S bit set to the value 0.

The possible values of this bit are:

0 The region is not shareable.

1 The region is shareable.

TR*k*, bits[15:0] Primary TEX mapping for memory attributes *m*, where *m* is the value of the TEX[0] C and B bits in the translation table entry. This field defines the mapped memory type for a region with attributes *n*.

The encoding of this field is:

`0b00` Strongly-ordered.

`0b01` Device.

`0b10` Normal memory.

`0b11` Reserved. Effect is UNPREDICTABLE.

The meaning of the field with *n* = 6 is IMPLEMENTATION DEFINED and might differ from the meaning given here. This is because the meaning of the attribute combination TEX[0] = 1, C = 1, B = 0 is IMPLEMENTATION DEFINED.

Table 15-5 shows the mapping between the memory region attributes and the value *n* used in the SMMU_CB*n*_PRRR.NOS*n* and SMMU_CB*n*_PRRR.TR*n* field descriptions.

Table 15-5 TEX[0] mappings

TEX[0]	C	B	*n* value
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

15.5.28 SMMU_CB*n*_RESUME, Transaction Resume register

The SMMU_CB*n*_RESUME characteristics are:

Purpose Resumes or terminates the operation of a stalled transaction.

This is the only mechanism for clearing the SMMU_CBn_FSR.SS bit.

If a stalled transaction is retried by writing 0 to this register, the retry always occurs before the first use of SMMU_CBn_TTBCR. For nested translations stalled in stage 2, the translation restarts in stage 1.

This means that a retried transaction is guaranteed to be affected by changes to the SMMU_CBn_TTBCR. However it is not guaranteed to be affected by changes to registers such as SMMU_SSDRn, SMMU_S2CRn, or SMMU_CBARn.

Note

Optionally, an implementation can retry a transaction from an earlier stage of the translation process. For example, transactions could be subject to security state determination again.

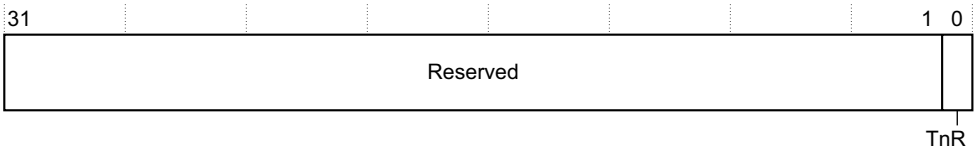
See *Handling a Context fault on page 3-53* for more information about writing to this register.

Usage constraints If the SMMU_CBn_SCTLR.CFCFG bit is set to 0, indicating the Terminate model, while there is a stalled transaction the result is UNPREDICTABLE.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit WO register.

The SMMU_CB*n*_RESUME bit assignments are:



Bits[31:1] Reserved.

TnR, bit[0] Terminate not Retry.

The possible values of this bit are:

0 Retry the stalled transaction.

1 Terminate the stalled transaction.

15.5.29 SMMU_CB*n*_SCTLR, System Control Register

The SMMU_CB*n*_SCTLR characteristics are:

Purpose Provides top-level control of the translation system for the related Translation context bank.

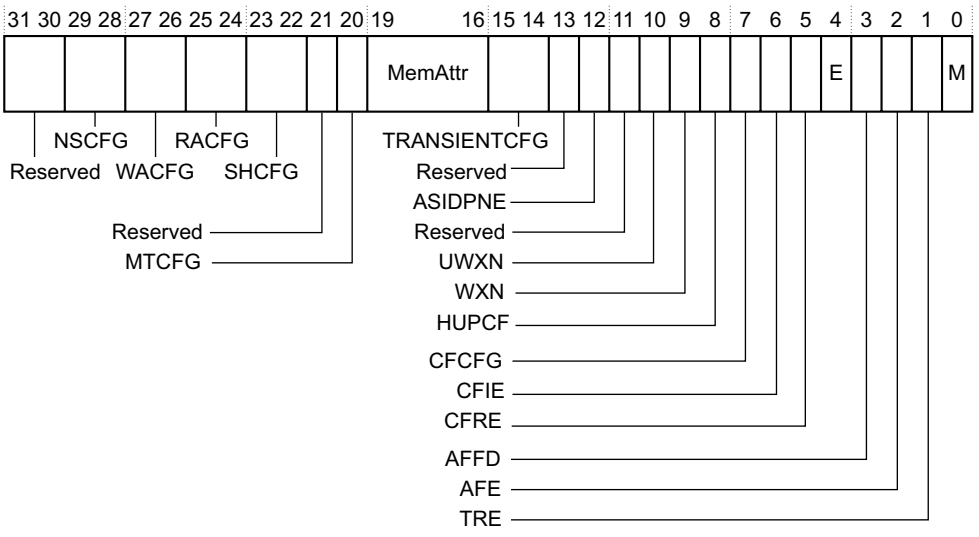
Usage constraints No usage constraints apply.

Configurations See the field descriptions for field-specific configuration details.

It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit RW register. With the exception of SMMU_CB*n*_SCTLR[6:5], the fields in this register have UNPREDICTABLE reset values.

The SMMU_CB*n*_SCTLR bit assignments are:



Bits [31:30] Reserved.

NSCFG, bits[29:28]

Non-Secure Configuration, controls the Non-secure attribute for any transaction where the Translation context bank translation is disabled. That is, where SMMU_CB*n*_SCTLR.M==0.

SMMU_CB*n*_SCTLR.NSCFG only exists in a Translation context bank reserved by Secure software. In a Non-secure Translation context bank, this field is UNK/SBZP.

The encoding of this field is:

`0b00` Default Non-secure attribute.

`0b01` Reserved.

`0b10` Secure.

`0b11` Non-secure.

WACFG, bits[27:26]

Write-Allocate Configuration, controls the allocation hint for a write transaction where the Translation context bank translation is disabled. That is, where SMMU_CB*n*_SCTLR.M==0.

The encoding of this field is:

`0b00` Default allocation attributes.

`0b01` Reserved.

`0b10` Write-Allocate.

`0b11` No Write-Allocate.

RACFG, bits[25:24]

Read-Allocate Configuration, controls the allocation hint for a read transaction where the Translation context bank translation is disabled. That is, where SMMU_CB*n*_SCTLR.M==0.

The encoding of this field is:

`0b00` Default allocation attributes.

`0b01` Reserved.

`0b10` Read-Allocate.

`0b11` No Read-Allocate.

SHCFG, bits[23:22]

Shared Configuration, controls the shareable attribute of a transaction where the Translation context bank is disabled. That is, where SMMU_CB*n*_SCTLR.M==0.

The encoding of this field is:

`0b00` Default shareable attribute.

`0b01` Outer Shareable.

`0b10` Inner Shareable.

`0b11` Non-shareable.

Bit[21] Reserved.

MTCFG, bit[20]

Memory Type Configuration, controls the memory type attribute of a transaction where the Translation context bank translation is disabled. That is, where SMMU_CB*n*_SCTLR.M==0.

The possible values of this bit are:

0 Default memory attributes.

1 Use the MemAttr field for memory attributes.

MemAttr, bits[19:16]

Memory Attributes. These attributes are permitted to be overlaid if SMMU_CB*n*_SCTLR.M==0 and SMMU_CB*n*_SCTLR.MTCFG==1.

Table 15-6 and Table 15-7 show the MemAttr field encoding.

Table 15-6 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 15-7 Secondary MemAttr bit values

Bits[1:0]	Meaning when bits[3:2] == 0b00	Meaning when bits[3:2] != 0b00
0b00	Strongly-ordered	Reserved
0b01	Device	Inner Non-cacheable Normal memory
0b10	Reserved	Inner Write-Through Normal memory
0b11	Reserved	Inner Write-Back Normal memory

TRANSIENTCFG, bits[15:14]

Transient Allocate Configuration, controls the transient allocation hint.

The encoding of this field is:

0x00 Default transient allocation attributes.

0x01 Reserved.

0x10 Non-transient.

0x11 Transient.

This field applies where SMMU_CBn_SCTLR.M== 0.

It is IMPLEMENTATION DEFINED whether this field is present. If not implemented, these bits behave as RAZ/WI.

Bit[13] Reserved.

ASIDPNE, bit[12]

Address Space Identifier Private Namespace Enable.

The possible values of this bit are:

0 The System MMU ASID values for this Translation context bank are coordinated with the wider system.

1 The System MMU ASID values for this Translation context bank are a private namespace that is not coordinated with the wider system.

If ASIDPNE==1, a broadcast TLB Invalidate operation specifying an ASID value is not required to apply to cached translations in the System MMU. This field is a hint. A Broadcast TLB Invalidate operation is still permitted to affect cached translation in the System MMU and is permitted to apply to all unlocked entries.

The scope of the ASID namespace for a Translation context bank is limited by the VMID associated with that Translation context bank in the stream to context mapping process.

Bit[11] Reserved.

UWXN, bit[10]

Unprivileged Writable Execute Never.

The possible values of this bit are:

0 This behavior is not enabled.

1 Raise a stage 1 permission fault if an instruction fetch occurs from a memory location that permits writes for unprivileged accesses.

Note

This field only applies to translated transactions. That is, when SMMU_CB*n*_SCTLR.M==1.

WXN, bit[9] Writable Execute Never.

The possible values of this bit are:

0 This behavior is not enabled.

1 Raise a stage 1 permission fault if an instruction fetch occurs from a memory location that permits writes.

Note

This field only applies to translated transactions. That is, when SMMU_CB*n*_SCTLR.M==1.

HUPCF, bit[8] Hit Under Previous Context Fault.

The possible values of this bit are:

0 Stall or terminate subsequent transactions in the presence of an outstanding context fault.

1 Process all subsequent transactions independently of any outstanding context fault.

CFCFG, bit[7] Context Fault Configuration.

The possible values of this bit are:

0 Terminate.

1 Stall.

CFIE, bit[6] 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 bit resets to 0.

CFRE, bit [5] 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.

This bit resets to 0.

For exclusive access transactions, when this bit is set to 0, ARM recommends that the SMMU reports the transaction as failed. See *Reporting exclusive access transactions on page 3-61* for more information.

E, bit[4] Endianness, indicates the endian format of translation table entries.

The possible values of this bit are:

0 Little endian format.

1 Big endian format.

AFFD, bit[3] Access Flag Fault Disable, determines whether Access flag faults are enabled. This bit only applies when AFE==1.

The possible values of this bit are:

0 Access flag faults are enabled.

1 Access flag faults are disabled.

If AFFD==0, AP[0]==0 in the translation table entry causes an Access flag fault, which is reported by SMMU_CBn_FSR.

If AFFD==1, hardware behaves as if AP[0]==1, regardless of the translation table entry value.

For more information about the *Access permission* (AP) bit, see the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition*.

AFE, bit[2] Access Flag Enable, enables the AP[0] bit in the translation table descriptors to be used as an access flag.

The possible values of this bit are:

0 In the translation table descriptors, AP[0] is an access permissions bit. The full range of access permissions is supported. No access flag is implemented.

1 In the translation table descriptors, AP[0] is an access flag. Only the simplified model for access permissions is supported.

If the Long-descriptor translation table format is enabled, that is, if SMMU_CBn_TTBCR.EAE==1, this bit has no effect and the System MMU behaves as if the bit is set. ARM recommends that software treats this bit as UNK/SBOP when SMMU_CBn_TTBCR.EAE==1.

TRE, bit[1] TEX Remap Enable, enables remapping of the TEX[2:1] bits for use as two translation table bits that can be managed by the operating system. Enabling this remapping also changes the scheme that describes the memory region attributes in the VMSA.

The possible values of this bit are:

0 TEX Remap is disabled. Bits TEX[2:0] are used with the C and B bits to describe the memory region attributes.

1 TEX Remap is enabled. Bits TEX[2:1] are reassigned for use as flags managed by the operating system. The TEX[0], C and B bits describe the memory region attributes, with the MMU remap registers.

If the Long-descriptor translation table format is enabled, that is, if SMMU_CBn_TTBCR.EAE has the value 1, this bit has no effect and the System MMU behaves as if the bit is set. ARM recommends that software treats this bit as UNK/SBOP when SMMU_CBn_TTBCR.EAE has the value 1.

M, bit[0] MMU Enable, 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.

15.5.30 SMMU_CB*n*_TLBIALL, TLB Invalidate All

The SMMU_CB*n*_TLBIALL characteristics are:

Purpose Invalidates all of the unlocked TLB entries that are tagged as:

- hypervisor, if the context bank is currently marked as hypervisor context
- Non-secure, using the VMID of the context bank, if the context bank is Non-secure, non-hypervisor context
- Secure, using any ASID, if the context bank is Secure

Optionally, in addition, an SMMU implementation can over-invalidate, removing any arbitrary set of unlocked TLB entries, including those allocated from other context banks.

Usage constraints This operation requires no arguments, and only has to apply to TLB entries associated with the VMID used for the Stage 1 Translation context bank.

If SMMU_CBARn.HYPC has the value 1, this operation only has to apply to TLB entries associated with hypervisor contexts. The VMID is therefore irrelevant to the operation. See *Hypervisor-marked contexts on page 2-41* for more information.

In an implementation that includes the Security Extensions, this operation only has to apply to TLB entries associated with the security domain that the Stage 1 Translation context bank is a member of. The VMID of Secure context banks is ignored, even in implementations that support a VMID for Secure banks. See *Interaction with the Security Extensions on page 2-32* for more information.

See *Memory-mapped TLB maintenance operations on page 5-75* for more details about the usage model.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit WO command. Reads are SBZ.

The SMMU_CB*n*_TLBIALL register bit assignments are reserved.

15.5.31 SMMU_CB*n*_TLBIASID, TLB Invalidate by ASID

The SMMU_CB*n*_TLBIASID characteristics are:

Purpose Invalidates all of the unlocked TLB entries that match the ASID provided as an argument.

Optionally, in addition, an SMMU implementation can over-invalidate, removing any arbitrary set of unlocked TLB entries, including those allocated from other context banks.

Usage constraints This operation only has to apply to unlocked non-global TLB entries that match the VMID used for the Stage 1 Translation context bank. If SMMU_CBARn.HYPC is 1, this operation is UNPREDICTABLE.

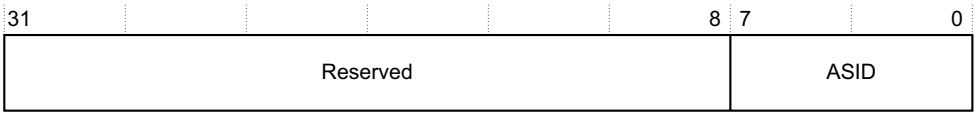
In addition, in an implementation that includes the Security Extensions, this operation only has to apply to unlocked TLB entries associated with the security domain that the Stage 1 Translation context bank is a member of.

See *Memory-mapped TLB maintenance operations on page 5-75* for more details about the usage model.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit WO command. Reads are SBZ.

The SMMU_CB*n*_TLBIASID bit assignments are:



Bits[31:8] Reserved.

ASID, bits[7:0]

Address Space Identifier, the input to the invalidation operation.

15.5.32 SMMU_CB*n*_TLBIVA, Invalidate TLB by VA

The SMMU_CB*n*_TLBIVA characteristics are:

Purpose Invalidates all of the unlocked TLB entries that match both the VA provided and the TLB tagging scheme of the context bank, including any global entries if appropriate.

Optionally, in addition, an SMMU implementation can over-invalidate, removing any arbitrary set of unlocked TLB entries, including those allocated from other context banks.

Usage constraints This operation only has to apply to unlocked TLB entries associated with the ASID and VMID used for a Stage 1 Translation context bank. The ASID is not checked for global entries in the TLB. The VMID of Secure context banks is ignored for TLB matching purposes, even in implementations that support a VMID for Secure banks. See *Interaction with the Security Extensions on page 2-32* for more information.

If SMMU_CBARn.HYPC has the value 1, this operation only has to apply to unlocked TLB entries associated with hypervisor contexts. The VMID and ASID are therefore irrelevant to the operation. See *Hypervisor-marked contexts on page 2-41* for more information.

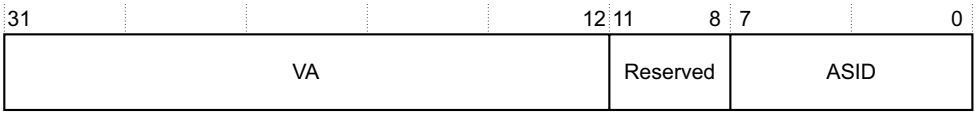
In an implementation that includes the Security Extensions, this operation is only required to apply to unlocked TLB entries associated with the security domain that the Stage 1 Translation context bank is a member of.

See *Memory-mapped TLB maintenance operations on page 5-75* for more details about the usage model.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit WO command. Reads are SBZ.

The SMMU_CB*n*_TLBIVA bit assignments are:



VA, bits[31:12]

Virtual Address, the input address to the invalidation operation.

Bits[11:8] Reserved.

ASID, bits[7:0]

Address Space Identifier, the input to the invalidate operation.

15.5.33 SMMU_CB*n*_TLBIVAA, TLB Invalidate by VA All ASID

The SMMU_CB*n*_TLBIVAA characteristics are:

Purpose Invalidates all of the unlocked TLB entries that match the VA provided as an argument, and the VMID of the context bank, regardless of the ASID. This operation includes global entries if appropriate.

Optionally, in addition, an SMMU implementation can over-invalidate, removing any arbitrary set of unlocked TLB entries, including those allocated from other context banks.

Usage constraints This operation only has to apply to unlocked TLB entries associated with the VMID used for a Stage 1 Translation context bank. If SMMU_CBARn.HYPC has the value 1, this operation is UNPREDICTABLE.

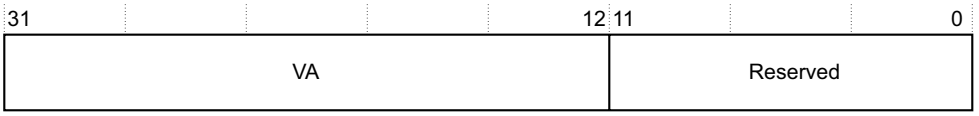
In an implementation that includes the Security Extensions, this command must operate on all unlocked TLB entries associated with the security domain that the Stage 1 Translation context bank is a member of. The VMID of Secure context banks is ignored for TLB matching purposes, even in implementations that support a VMID for Secure banks. See *Interaction with the Security Extensions on page 2-32* for more information.

See *Memory-mapped TLB maintenance operations on page 5-75* for more details about the usage model.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit WO command. Reads are SBZ.

The SMMU_CB*n*_TLBIVAA bit assignments are:



VA, bits[31:12]

Virtual Address, the input address to the invalidate operation.

Bits[11:0] Reserved.

15.5.34 SMMU_CB*n*_TLBIVAAL, TLB Invalidate by VA, All ASID, Last level

The SMMU_CB*n*_TLBIVAAL characteristics are:

Purpose Invalidates all of the unlocked TLB entries that match the VA provided as an argument, and the VMID of the context bank, regardless of the ASID. This operation includes global entries if appropriate.

Optionally, in addition, an SMMU implementation can over-invalidate, removing any arbitrary set of unlocked TLB entries, including those allocated from other context banks.

Usage constraints This command is similar to SMMU_CB*n*_TLBIVAA, but it is only required to invalidate cached copies of the last level of translation table walk of the first stage of translation. The operation only has to apply to unlocked TLB entries associated with the VMID used for a Stage 1 Translation context bank. The ASID is not checked for global entries in the TLB.

If SMMU_CBARn.HYPC has the value 1, this operation is UNPREDICTABLE.

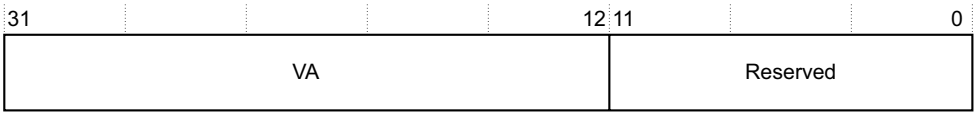
In an implementation that includes the Security Extensions, this command must operate on all unlocked TLB entries associated with the security domain that the Stage 1 Translation context bank is a member of. The VMID of Secure context banks is ignored for TLB matching purposes, even in implementations that support a VMID for Secure banks. See *Interaction with the Security Extensions on page 2-32* for more information.

See *Memory-mapped TLB maintenance operations on page 5-75* for more details about the usage model.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS.

Attributes A 32-bit WO command. Reads are SBZ.

The SMMU_CB*n*_TLBIVAAL bit assignments are:



VA, bits[31:12]

Virtual Address. This is the input address to the invalidate operation.

Bits[11:0] Reserved.

15.5.35 SMMU_CB*n*_TLBIVAL, TLB Invalidate by VA, Last level

The SMMU_CB*n*_TLBIVAL characteristics are:

Purpose Invalidates all of the unlocked TLB entries that match the VA and ASID provided as arguments, and the VMID of the context bank.

Optionally, in addition, an SMMU implementation can over-invalidate, removing any arbitrary set of unlocked TLB entries, including those allocated from other context banks.

Usage constraints This command is similar to SMMU_CB*n*_TLBIVA, but it is only required to invalidate cached copies of the last level of translation table walk of the first stage of translation. The operation only has to apply to TLB entries associated with the VMID used for a Stage 1 Translation context bank. The ASID is not checked for global entries in the TLB.

If SMMU_CBARn.HYPC has the value 1, this operation only has to apply to TLB entries associated with hypervisor contexts. VMID and ASID are therefore irrelevant to the operation. See *Hypervisor-marked contexts on page 2-41* for more information.

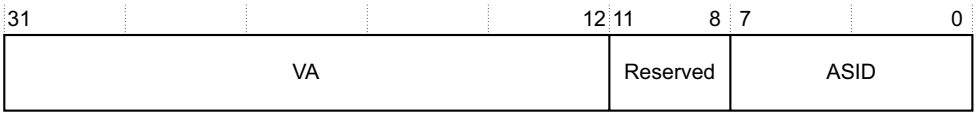
In an implementation that includes the Security Extensions, this command must operate on all unlocked TLB entries associated with the security domain that the Stage 1 Translation context bank is a member of. The VMID of Secure context banks is ignored, even in implementations that support a VMID for Secure banks. See *Interaction with the Security Extensions on page 2-32* for more information.

See *Memory-mapped TLB maintenance operations on page 5-75* for more details about the usage model.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit WO command. Reads are SBZ.

The SMMU_CB*n*_TLBIVAL bit assignments are:



VA, bits[31:12]

Virtual Address, the input address to the invalidation operation.

Bits[11:8] Reserved.

ASID, bits[7:0]

Address Space Identifier, the input ASID to the invalidation operation.

15.5.36 SMMU_CB*n*_TLBSTATUS, TLB Status

The SMMU_CB*n*_TLBSTATUS characteristics are:

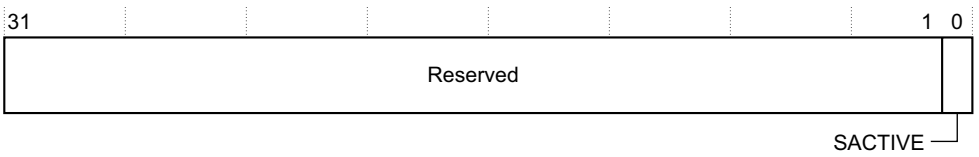
Purpose Invalidates the status of TLB maintenance operations.

Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit RO register. Writes are ignored.

The SMMU_CB*n*_STATUS bit assignments are:



Bits[31:1] Reserved.

SACTIVE, bit[0]

Synchronize TLB Invalidate Active.

The possible values of this bit are:

0 There is no active TLBsynchronization operation.

1 There is an active TLBsynchronization operation.

15.5.37 SMMU_CB*n*_TLBSYNC, TLB Synchronize Invalidate

The SMMU_CB*n*_TLBSYNC characteristics are:

Purpose Initiates a synchronization operation that ensures the completion of any TLB invalidate operations previously accepted in the corresponding Translation context bank.

Usage constraints This operation operates in the scope of the Translation context bank it resides in. After being accepted, the operation does not complete until all Translation context bank TLB invalidate operations accepted by the System MMU before the synchronize operation was accepted are complete.

See *Memory-mapped TLB maintenance operations on page 5-75* for more details about the usage model.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS.

Attributes A 32-bit WO command. Reads are SBZ.

The SMMU_CB*n*_TLBSYNC bit assignments are reserved.

15.5.38 SMMU_CB*n*_TTBCR, Translation Table Base Control Register

The SMMU_CB*n*_TTBCR characteristics are:

Purpose Determines which of the Translation Table Base Registers, SMMU_CBn_TTBRm, defines the base address for the translation table walk required when an input address is not found in the TLB.

Usage constraints No usage constraints apply.

Configurations The format of this register depends on the value of SMMU_CB*n*_TTBCR.EAE. For more information, see:

- *Register format when SMMU_CBn_TTBCR.EAE is 0*
- *Register format when SMMU_CBn_TTBCR.EAE is 1 on page 15-222*.

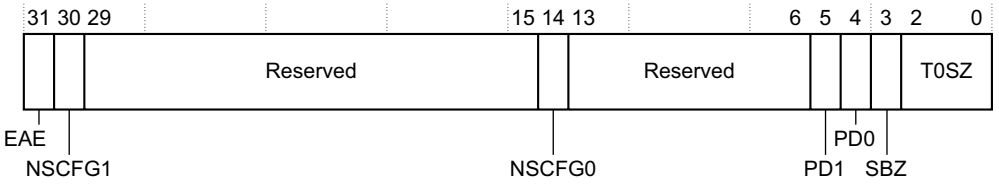
See also *Multi-format registers and reserved fields on page 15-193*.

It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes A 32-bit RW register with an UNPREDICTABLE reset value.

Register format when SMMU_CBn_TTBCR.EAE is 0

The SMMU_CB*n*_TTBCR.EAE bit assignments in this format are:



EAE, bit[31] Extended Address Enable.

The possible values of this bit are:

0 Use the translation system defined in the Short-descriptor.

1 Use the translation system defined in the Long-descriptor.

NSCFG1, bit[30]

Non-secure attribute for the memory associated with a translation table walk using SMMU_CB*n*_TTBR1. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* for more information.

This field only applies to a Secure Translation context bank. Otherwise, it is ignored.

Bits[29:15] Reserved.

NSCFG0, bit[14]

Non-secure attribute for the memory associated with a translation table walk using SMMU_CB*n*_TTBR0. This field only applies to a Secure Translation context bank. Otherwise, it is ignored. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224*.

Bits[13:6] Reserved.

PD1, bit[5] Unchanged from ARMv7-A.

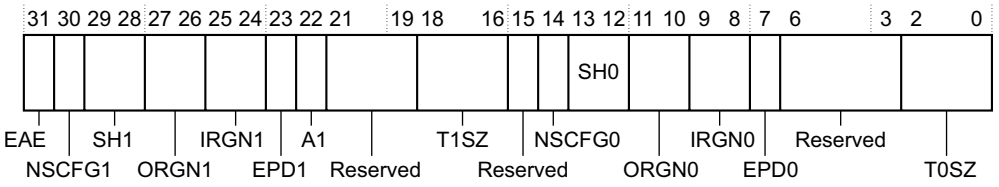
PD0, bit[4] Unchanged from ARMv7-A.

SBZ, bit[3] Should-be-Zero.

T0SZ, bits[2:0] The size offset of the SMMU_CB*n*_TTBR0 addressed region, encoded as a 3-bit unsigned number giving the size of the region as $2^{32-T0SZ}$. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224*.

Register format when SMMU_CBn_TTBCR.EAE is 1

The SMMU_CB*n*_TTBCR.EAE bit assignments in this format are:



EAE Extended Address Enable.

The possible values of this bit are:

0 Use the translation system defined in the Short-descriptor

1 Use the translation system defined in the Long-descriptor.

NSCFG1, bit[30]

Non-secure attribute for the memory associated with a translation table walk using SMMU_CB*n*_TTBR1. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* for more information.

This field only applies to a Secure Translation context bank. Otherwise, it is ignored.

SH1, bits [29:28]

Shareable attributes for the memory associated with the translation table walks using SMMU_CB*n*_TTBR1. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* and *SH1, SH0 encoding on page 15-224* for more information.

ORGN1, bits [27:26]

Outer Cacheability Attributes for the memory associated with the translation table walks using SMMU_CB*n*_TTBR1. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* and *ORGN1, ORGN0, IRGN1, IRGN0 encoding on page 15-224* for more information.

IRGN1, bits [25:24]

Inner Cacheability Attributes for the memory associated with the translation table walks using SMMU_CB*n*_TTBR1. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* and *ORGN1, ORGN0, IRGN1, IRGN0 encoding on page 15-224* for more information.

EPD1, bit[23] Translation Walk Disable for the SMMU_CB*n*_TTBR1 region where SMMU_CB*n*_TTBCR.EAE==1. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* for more information. Controls whether a translation table walk is performed on a TLB miss when SMMU_CB*n*_TTBR1 is used.

This bit is UNK/SBZ if either:

- SMMU_CB*n*_TTBCR.EAE==0
- SMMU_CBARn.HYPC==1.

The possible values of this bit are:

0 If a TLB miss occurs when SMMU_CB*n*_TTBR1 is used, a translation table walk is performed.

1 If a TLB miss occurs when SMMU_CB*n*_TTBR1 is used, no translation table walk is performed and an L1 Section translation fault is returned.

This function is the same as PD1. The bit position is moved depending on the setting of SMMU_CB*n*_TTBCR.EAE.

A1, bit[22] Select the ASID from the SMMU_CB*n*_TTBR1 or SMMU_CB*n*_TTBR0 ASID field. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* for more information.

The possible values of this bit are:

0 Select the ASID from the SMMU_CB*n*_TTBR0 ASID field.

1 Select the ASID from the SMMU_CB*n*_TTBR1 ASID field.

Bits[21:19] Reserved.

T1SZ, bits[18:16]

The size offset of the SMMU_CB*n*_TTBR1 addressed region, encoded as a 3-bit unsigned number, giving the size of the region as $2^{32-T1SZ}$. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224*.

Bit[15] Reserved.

NSCFG0, bit[14]

Non-secure attribute for the memory associated with a translation table walk using SMMU_CB*n*_TTBR0. This field only applies to a Secure Translation context bank. Otherwise, it is ignored. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224*.

SH0, bits[13:12]

Shareable attributes for the memory associated with the translation table walks using SMMU_CB*n*_TTBR0. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* and *SH1, SH0 encoding on page 15-224* for more information.

ORGN0, bits[11:10]

Outer Cacheability Attributes for the memory associated with the translation table walks using SMMU_CB*n*_TTBR0. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* and *ORGN1, ORGN0, IRGN1, IRGN0 encoding on page 15-224* for more information.

IRGN0, bits[9:8]

Inner Cacheability Attributes for the memory associated with the translation table walks using SMMU_CB*n*_TTBR0. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* and *ORGN1, ORGN0, IRGN1, IRGN0 encoding on page 15-224* for more information.

EPD0, bit[7] Translation Walk Disable for the SMMU_CB*n*_TTBR0 region, where EAE==1. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* for more information. Controls whether a translation table walk is performed on a TLB miss when SMMU_CB*n*_TTBR0 is used.

This bit is UNK/SBZ if either:

- SMMU_CB*n*_TTBCR.EAE==0
- SMMU_CBARn.HYPC==1.

The possible values of this bit are:

0 If a TLB miss occurs when SMMU_CB*n*_TTBR0 is used, a translation table walk is performed.

1 If a TLB miss occurs when SMMU_CB*n*_TTBR0 is used, no translation table walk is performed and an L1 Section translation fault is returned.

This function is the same as PD0. The bit position is moved depending on the setting of SMMU_CB*n*_TTBCR.EAE.

Bits[6:3] Reserved.

T0SZ, bits[2:0]

The size offset of the SMMU_CB*n*_TTBR0 addressed region, encoded as a 3-bit unsigned number, giving the size of the region as $2^{32-T0SZ}$. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224*.

SH1, SH0 encoding

Table 15-8 shows the encoding of the SH1 and SH0 fields.

Table 15-8 SMMU_CB*n*_TTBCR.SH1 & SH0 encoding

SH[1]	SH[0]	Normal memory
0	0	Non-shareable
0	1	UNPREDICTABLE
1	0	Outer Shareable
1	1	Inner Shareable

ORGN1, ORGN0, IRGN1, IRGN0 encoding

Table 15-9 shows the encoding of the ORGN1, ORGN0, IRGN1 and IRGN0 fields.

Table 15-9 SMMU_CB*n*_TTBCR.ORGN1, ORGN0, IRGN1, IRGN0 encoding

RGN[1]	RGN[0]	Description
0	0	Non-cacheable Normal memory
0	1	Write-back, Write-Allocate cacheable
1	0	Write-Through cacheable
1	1	Write-back, no Write-Allocate cacheable

15.5.39 SMMU_CB*n*_TTBR*m*, Translation Table Base Registers

The SMMU_CB*n*_TTBR1 and SMMU_CB*n*_TTBR0 characteristics are:

Purpose Holds the translation table base addresses:
- SMMU_CB*n*_TTBR1 holds the base address of translation table 1.
- SMMU_CB*n*_TTBR0 holds the base address of translation table 0.

Usage constraints No usage constraints apply.

Configurations Two formats of Translation Table Base Register can be selected, the legacy Short-descriptor translation table format, and the more recent Long-descriptor translation table format. For more information, see:
- *Short-descriptor translation table format on page 15-225*
- *Long-descriptor translation table format on page 15-226*.

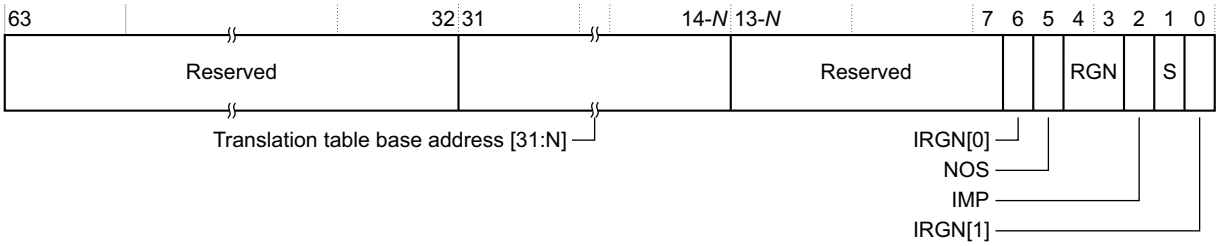
See also *Multi-format registers and reserved fields on page 15-193*.

It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes 64-bit RW registers with UNPREDICTABLE reset values.

Short-descriptor translation table format

When SMMU_CBn_TTBCR.EAE==0, the translation table base register bit assignments are:



Bits[63:32] Reserved.

Translation table base address [31:*N*]

Translation table base 1 address or translation table base 0 address, bits [31:14-*N*], where *N* determines the required alignment of the translation table, which must be aligned to 2^{14-N} bytes:

- For SMMU_CB*n*_TTBR1, *N*==0.
- For SMMU_CB*n*_TTBR0, *N* is determined by SMMU_CBn_TTBCR.T0SZ.

Bits[13-N:7] Reserved.

IRGN[0], bit[6]

Inner Region bits. See also SMMU_CB*n*_TTBR*m*.IRGN[1].

These bits indicate the Inner cacheability attributes for the memory associated with the translation table walks.

The encoding of bits[0,6] is:

`0b00` Inner Non-cacheable Normal memory.

`0b01` Inner Write-Back Write-Allocate cacheable Normal memory.

`0b10` Inner Write-Through cacheable Normal memory.

`0b11` Inner Write-Back no Write-Allocate cacheable Normal memory.

NOS, bit[5] Not Outer Shareable bit.

This bit indicates the Outer Shareable attribute for the memory associated with a translation table walk that has the Shareable attribute, indicated by SMMU_CB*n*_TTBR1.S or SMMU_CB*n*_TTBR0.S having the value 1.

The possible values of this bit are:

0 Outer Shareable.

1 Inner Shareable.

RGN, bits[4:3] Region bits, indicate the Outer cacheable attributes for the memory associated with the translation table walk.

The encoding of this field is:

`0b00` Non-cacheable Normal memory.

`0b01` Outer Write-Back Write-Allocate cacheable.

`0b10` Outer Write-Through cacheable.

`0b11` Outer Write-Back no Write-Allocate cacheable.

IMP, bit[2] IMPLEMENTATION DEFINED bit.

The effect of this bit is IMPLEMENTATION DEFINED. If the translation table implementation does not include any IMPLEMENTATION DEFINED features, this bit is SBZ.

S, bit[1] Shareable bit, indicates the Shareable attribute for the memory associated with a translation table walk.

The possible values of this bit are:

0 Non-shareable.

1 Shareable.

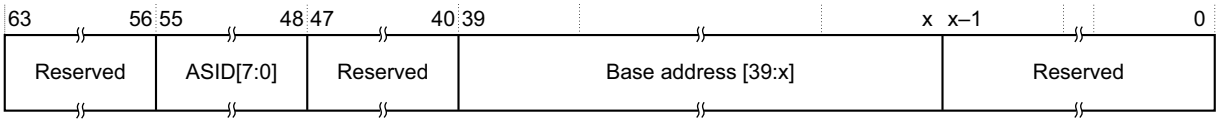
IRGN[1], bit[0]

Inner region bits. See also SMMU_CBn_TTBR*m*.IRGN[0].

These bits indicate the Inner cacheability attributes for the memory associated with a translation table walk.

Long-descriptor translation table format

When SMMU_CBn_TTBCR.EAE==1, the translation table base register bit assignments are:



Bits[63:56] Reserved.

ASID[7:0], bits[55:48]

The Address Space Identifier associated with this base address.

SMMU_CBn_TTBCR.A1 determines the selection between SMMU_CB*n*_TTBR0.ASID and SMMU_CB*n*_TTBR1.ASID.

Bits[47:40] Reserved.

Base Address [39:x]

Translation table base 1 or 0 address, bits [39:*x*].

The value *x* is determined by the T0SZ or T1SZ field in SMMU_CBn_TTBCR, according to the following formula, where *n* is 0 or 1 depending on whether T0SZ or T1SZ is specified:

```
if SMMU_CBn_TTBCR.TnSZ > 1 then
   x = 14 - SMMU_CBn_TTBCR.TnSZ
else
   x = 5 - SMMU_CBn_TTBCR.TnSZ
```

Bits[*x*-1:0] Reserved.

Chapter 16
Stage 2 Translation Context Bank Format

This chapter gives the layout of the Stage 2 Translation context bank. It contains the following sections:

16.1 Stage 1 and Stage 2 context bank format differences

The format of a Stage 2 Translation context bank is similar to a Stage 1 Translation context bank. The following differences apply to the stage 2 format:

- Registers that do not exist and are UNK/SBZP:
  - SMMU_CB*n*_NMRR
  - SMMU_CB*n*_PRRR
  - SMMU_CB*n*_TTBR1
  - SMMU_CB*n*_CONTEXTIDR.
- Registers that do not exist:
  - SMMU_CB*n*_ATSR
  - SMMU_CB*n*_MAIR1, SMMU_CB*n*_MAIR0
  - SMMU_CB*n*_PAR
  - SMMU_CB*n*_TLBSTATUS.
- Commands that do not exist:
  - SMMU_CB*n*_ATS1PR
  - SMMU_CB*n*_ATS1PW
  - SMMU_CB*n*_ATS1UR
  - SMMU_CB*n*_ATS1UW
  - SMMU_CB*n*_TLBIALL
  - SMMU_CB*n*_TLBIASID
  - SMMU_CB*n*_TLBIVA
  - SMMU_CB*n*_TLBIVAA
  - SMMU_CBn_TLBIVAAL
  - SMMU_CBn_TLBIVAL
  - SMMU_CB*n*_TLBSYNC.
- A Stage 2 Translation context bank only supports the LPAE mode of operation. This means that SMMU_CBn_TTBCR.EAE has a fixed value of 1.
- The fields in SMMU_CBn_TTBCR that relate to SMMU_CB*n*_TTBR1 are UNK/SBZP. See *SMMU_CBn_TTBRm, Translation Table Base Registers on page 15-224* for more information.

16.2 Stage 2 Translation context bank address space

The Stage 2 Translation context bank address space is defined in terms of the offset from its Translation context bank base address, *SMMU_CBn_BASE*, as Table 16-1 shows.

See also *Stage 1 and Stage 2 context bank format differences on page 16-228*.

Table 16-1 Stage 2 Translation context bank address space

Offset	Name	Type	Description	Notes
0x00000	SMMU_CBn_SCTLR	RW	*SMMU_CBn_SCTLR, System Control Register on page 16-234*.	-
0x00004	SMMU_CBn_ACTLR	RW	*SMMU_CBn_ACTLR, Auxiliary Control Register on page 15-194*.	-
0x00008	SMMU_CBn_RESUME	WO	*SMMU_CBn_RESUME, Transaction Resume register on page 15-210*.	-
0x0000C-0x0001C	Reserved	-	-	-
0x00020	SMMU_CBn_TTBR0[31:0]		*SMMU_CBn_TTBR0, Translation Table Base Register on page 16-238*.	-
0x00024	SMMU_CBn_TTBR0[63:32]	RW[a]	64-bit, *SMMU_CBn_TTBR0, Translation Table Base Register on page 16-238*.	64-bit
0x00028-0x0002C	Reserved	-	-	-
0x00030	SMMU_CBn_TTBCR	RW	*SMMU_CBn_TTBCR, Translation Table Base Control Register on page 16-237*.	-
0x00034-0x0003C	Reserved	-	-	-
0x00040-0x00044	Reserved	-	Reserved for IMPLEMENTATION DEFINED memory attributes associated with memory encodings. Such attributes are only additional qualifiers to the memory locations, and cannot change the architected behaviors of the memory attributes defined in SMMU_CB*n*_MAIR0 and SMMU_CB*n*_MAIR1. See *SMMU_CBn_MAIRm, Memory Attribute Indirection Registers on page 15-201* for more information.	
0x0004C-0x00054	Reserved	-	-	-
0x00058	SMMU_CBn_FSR	RW	*SMMU_CBn_FSR, Fault Status Register on page 15-197*.	-
0x0005C	SMMU_CBn_FSRRESTORE	WO	*SMMU_CBn_FSRRESTORE, Fault Status Restore Register on page 15-198*.	-
0x00060	SMMU_CBn_FAR[31:0]	RW	*SMMU_CBn_FAR, Fault Address Register on page 15-197*.	-
0x00064	SMMU_CBn_FAR[63:32]			
0x00068	SMMU_CB*n*_FSYNR0	RW	*SMMU_CBn_FSYNRm, Fault Syndrome Registers on page 15-199*.	-
0x0006C	SMMU_CB*n*_FSYNR1			
0x00070-0x00CFC	Reserved	-	-	-

Table 16-1 Stage 2 Translation context bank address space (continued)

Offset	Name	Type	Description	Notes
0x00DFC-0x00D00	IMPLEMENTATION DEFINED	RW	Reserved for IMPLEMENTATION DEFINED purposes.	-
0x00E00-0x00E38	SMMU_CBn_PMEVCNTRm	RW	*SMMU_CBn_PMEVCNTRm, Performance Monitors Event Counter registers on page 15-207*.	-
0x00E3C-0x00E7C	Reserved	-	-	-
0x00E80-0x00EBC	SMMU_CBn_PMEVTYPERm	RW	*SMMU_CBn_PMEVTYPERm, Performance Monitors Event Type Registers on page 15-207*.	-
0x00EBC-0x00EFC	Reserved	-	-	-
0x00F00	SMMU_CBn_PMCFGR	RO	*SMMU_CBn_PMCFGR, Performance Monitors Configuration Register on page 15-206*.	-
0x00F04	SMMU_CBn_PMCR	RW	*SMMU_CBn_PMCR, Performance Monitors Control Register on page 15-207*.	-
0x00F08-0x00F1C	Reserved	-	-	-
0x00F20	SMMU_CB*n*_PMCEID0	RO	*SMMU_CBn_PMCEIDm, Performance Monitors Common Event Identification registers on page 15-206*.	-
0x00F24	SMMU_CB*n*_PMCEID1	RO	*SMMU_CBn_PMCEIDm, Performance Monitors Common Event Identification registers on page 15-206*.	-
0x00F28-0x00F3C	Reserved	-	-	-
0x00F40	SMMU_CBn_PMCNTENSET	RW	*SMMU_CBn_PMCNTENSET, Performance Monitors Count Enable Set register on page 15-207*.	-
0x00F44	SMMU_CBn_PMCNTENCLR	RW	*SMMU_CBn_PMCNTENCLR, Performance Monitors Count Enable Clear register on page 15-206*.	-
0x00F48	SMMU_CBn_PMINTENSET	RW	*SMMU_CBn_PMINTENSET, Performance Monitors Interrupt Enable Set register on page 15-208*.	-
0x00F4C	SMMU_CBn_PMINTENCLR	RW	*SMMU_CBn_PMINTENCLR, Performance Monitors Interrupt Enable Clear register on page 15-208*.	-
0x00F50	SMMU_CBn_PMOVSCLR	RW	*SMMU_CBn_PMOVSCLR, Performance Monitors Overflow Status Clear Register on page 15-208*.	-
0x00F54	Reserved	-	-	-
0x00F58	SMMU_CBn_PMOVSSET	RW	*SMMU_CBn_PMOVSSET, Performance Monitors Overflow Status Set Register on page 15-208*.	-

Table 16-1 Stage 2 Translation context bank address space (continued)

Offset	Name	Type	Description	Notes
0x00F5C-0x00FB4	Reserved	-	-	-
0x00FB8	SMMU_CBn_PMAUTHSTATUS	RO	*SMMU_CBn_PMAUTHSTATUS, Performance Monitors Authentication Status register on page 15-206*.	-
0x00FBC-0x00FC8	Reserved	-	-	-
0x00FCC	Reserved		-	-
0x00FD0-0x00FFC	IMPLEMENTATION DEFINED	-	-	Reserved for Primecell ID
0x01000-(*PAGESIZE* − 0x4)	Reserved	-	-	-

a. Mainly RW. See field descriptions for detail.

16.3 Stage 2 Translation context bank register descriptions

This section describes some of the Stage 2 Translation context bank registers that might be present in a System MMU implementation. The registers described in this section differ slightly to those in the Stage 1 Translation context bank address space. See Table 16-1 on page 16-229 for the full Stage 2 Translation context bank address space map.

16.3.1 SMMU_CB*n*_FSYNR*m*, Fault Syndrome Registers

The SMMU_CB*n*_FSYNR1 and SMMU_CB*n*_FSYNR0 characteristics are:

Purpose Holds fault syndrome information about the memory access that caused a synchronous abort exception.

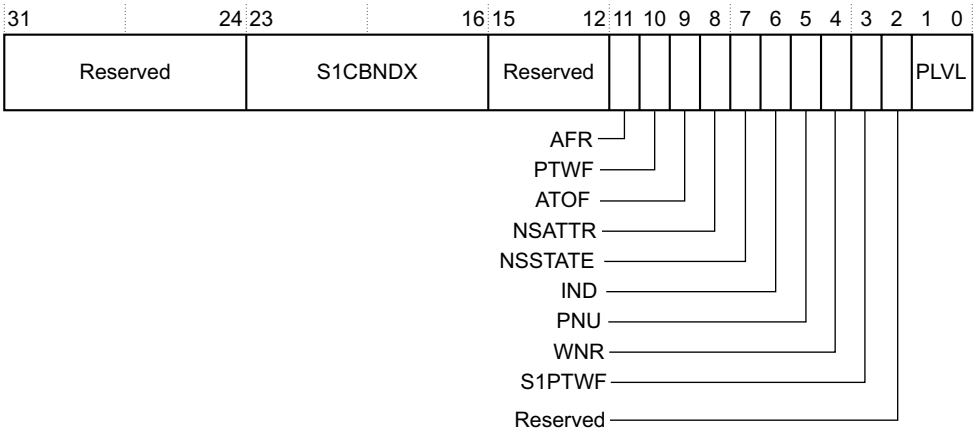
Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 1 translation. See SMMU_IDR0.S1TS for more information.

Attributes 32-bit RW registers with UNPREDICTABLE reset values.

SMMU_CB*n*_FSYNR0

The SMMU_CB*n*_FSYNR0 bit assignments are:



Bits[31:24] Reserved.

S1CBNDX[23:16]

Stage 1 Context Bank Index associated with the transaction that caused the fault.

For nested translation, this field contains the Stage 1 Translation context bank index for processing the transaction.

For stage 2 only translation, this field is UNKNOWN.

This field is only present in a stage 2 format Translation context bank. In a stage 1 format Translation context bank, it is UNK/SBPZ.

This field is only valid if SMMU_IDR0.NTS==1. In an implementation that does not include nested translation, this field is UNK/WI.

Bits[15:12] Reserved.

AFR, bit[11] Asynchronous Fault Recorded. The possible values of this bit are:

0 A fault was recorded synchronously.

1 A fault was recorded asynchronously.

PTWF, bit[10] A walk fault on a translation table access. The possible values of this bit are:

0 A walk fault did not occur.

1 A fault occurred during processing of a translation table walk.

ATOF, bit[9] Address Translation Operation Fault. The possible values of this bit are:

0 An ATOF fault did not occur.

1 A fault occurred during the processing of an address translation operation.

NSATTR, bit[8]

Non-secure Attribute. The possible values of this bit are:

0 The input transaction has a Secure attribute.

1 The input transaction has a Non-secure attribute.

NSSTATE, bit[7]

Non-secure State. The possible values of this bit are:

0 The transaction is associated with a Secure client.

1 The transaction is associated with a Non-secure client.

IND, bit[6] Instruction Not Data. The possible values of this bit are:

0 Data.

1 Instruction.

PNU, bit[5] Privileged Not Unprivileged. The possible values of this bit are:

0 Unprivileged.

1 Privileged.

WNR, bit[4] Write Not Read. The possible values of this bit are:

0 Read.

1 Write.

S1PTWF, bit[3]

A walk fault on a stage 1 translation table access. The possible values of this bit are:

0 A fault did not occur during stage 2 translation of a stage 1 translation table walk.

1 A fault occurred during stage 2 translation of a stage 1 translation table walk.

This field is only valid if the implementation supports nested translation. That is, if SMMU_IDR0.NTS==1. In an implementation that does not include nested translation, this field is UNK/SBZP.

Bit[2] Reserved.

PLVL, bits[1:0]

Translation Table Level, the level in the translation table walk that the fault is associated with. The encoding of this field is:

`0b01` Level 1.

`0b10` Level 2.

`0b11` Level 3.

Faults and translation table level association

The translation table level a fault is associated with is:

- For a fault associated with a translation table walk, the level of table walk being performed.
- For a translation fault, the level of translation table that gave the fault. If a disabled translation table walk causes the fault or if the size of the address presented is out of the range specified for matching with any base address register, the fault is reported for level 1.
- For an access fault, the level of translation table that gave the fault.

- For a permission fault, including a fault caused by a hierarchical permission, the final level of translation table used for that translation.

SMMU_CB*n*_FSYNR1

The 32-bit SMMU_CB*n*_FSYNR1 register bit assignments are IMPLEMENTATION DEFINED.

16.3.2 SMMU_CB*n*_SCTLR, System Control Register

The SMMU_CB*n*_SCTLR characteristics are:

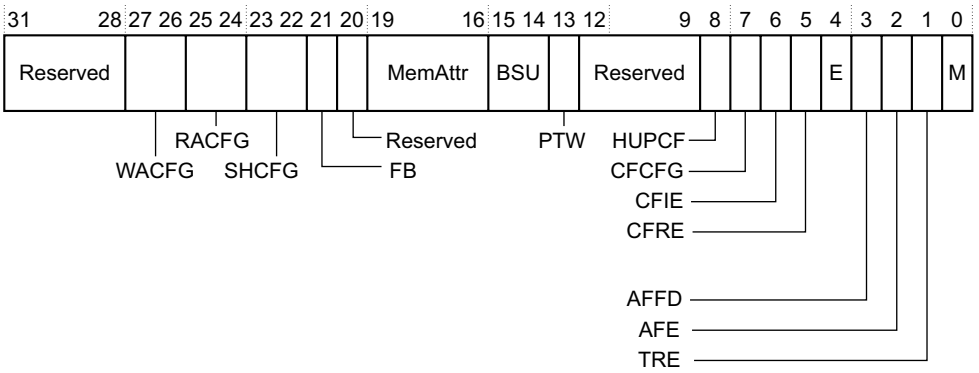
Purpose Provides top-level control of the translation system for Stage 2 Translation context bank *n*.

Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 2 translation. See:
- SMMU_IDR0.S2TS
- SMMU_IDR0.NTS
- *Translation context bank on page 2-27*.

Attributes A 32-bit RW register.

The SMMU_CB*n*_SCTLR bit assignments are:



Bits [31:28] Reserved.

WACFG, bits[27:26]

Write Allocate Configuration, controls the allocation hint for a write transaction where the Translation context bank translation is disabled. That is, where SMMU_CB*n*_SCTLR.M==0.

The encoding of this field is:

`0b00` Default allocation attributes.

`0b01` Reserved.

`0b10` Write-Allocate.

`0b11` No Write-Allocate.

RACFG, bits[25:24]

Read Allocate Configuration, controls the allocation hint for read transactions where the Translation context bank translation is disabled. That is, where SMMU_CB*n*_SCTLR.M==0.

The encoding of this field is:

`0b00` Default allocation attributes.

`0b01` Reserved.

`0b10` Read-Allocate.

`0b11` No Read-Allocate.

SHCFG, bits[23:22]

Shared Configuration, controls the shareable attributes of a transaction where the Translation context bank is disabled. That is, where SMMU_CB*n*_SCTLR.M==0.

The encoding of this field is:

`0b00` Reserved.

`0b01` Outer Shareable.

`0b10` Inner Shareable.

`0b11` Non-shareable.

This field differs from the equivalent field in the stage 1 format SMMU_CB*n*_SCTLR register. The stage 2 SMMU_CB*n*_SCTLR.SHCFG field is combined with the shared attributes of the previous translation step. See Table 11-2 on page 11-158 for more information.

FB, bit[21] Force Broadcast, forces the Broadcast of TLB maintenance, BPIALL and ICIALLU operations.

Bit[20] Reserved.

MemAttr, bits[19:16]

Memory Attributes.

The memory attributes are permitted to be overlaid if SMMU_CB*n*_SCTLR.M==0.

Table 16-2 and Table 16-3 show valid values for this field.

Table 16-2 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 16-3 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

This field differs from the equivalent field in the stage 1 SMMU_CB*n*_SCTLR format. The stage 2 MemAttr field is combined with the memory attributes presented from the previous translation step. See Table 11-2 on page 11-158.

BSU, bits[15:14]

Barrier Shareability Upgrade, 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 encoding of this field is:

`0b00` No effect.

`0b01` Inner Shareable.

`0b10` Outer Shareable.

`0b11` Full system.

The upgrade of the barrier shareability domain might not be supported in all system topologies. In an implementation that does not have this upgrade behavior, this field is RAZ/SBZP.

PTW, bit[13] Protected Translation Walk, only valid for an implementation that has nested translation. In an implementation that does not have nested translation, it is RAZ/SBZP.

The possible values of this bit are:

0 This behavior is not enabled.

1 Raise a Stage 2 permission fault if a stage 1 translation walk is to an area of memory that has the Device or Strongly-ordered memory attribute in the stage 2 translation tables.

Bit[12] Reserved.

The Stage 2 Translation context bank format does not provide the ASIDPNE field that exists at this location in the Stage 1 Translation context bank format. For a stage 2 format Translation context bank, this field is UNK/SBZP.

Bits[11:9] Reserved.

The Stage 2 Translation context bank format does not provide the WXN and UWXN fields. For a stage 2 format Translation context bank, these fields are UNK/SBZP.

HUPCF, bit[8] Hit Under Previous Context Fault.

The possible values of this bit are:

0 Stall or terminate any subsequent transaction in the presence of an outstanding context fault.

1 Process any subsequent transaction independently of any outstanding context fault.

The distributed nature of some SMMU implementations means there can be significant delays between detection of a context fault for one upstream client device and transactions from other upstream client devices being stopped.

When this bit is set to 0, it is not guaranteed that faults from different devices are recorded in absolute temporal order. When a device faults, although the setting of this bit is obeyed for that device, no temporal guarantees are made regarding suspension of transactions from different non-faulting devices.

CFCFG, bit[7] Context Fault Configuration.

The possible values of this bit are:

0 Terminate the transaction.

1 Stall the transaction.

CFIE, bit[6] 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 0.

CFRE, bit [5] 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.

E, bit[4] Endianness, indicates the endianness of translation table entries.

The possible values of this bit are:

0 Little endian format.

1 Big endian format.

AFFD, bit[3] Access Flag Fault Disable, determines whether Access flag faults are enabled. Only applicable when AFE==1.

The possible values of this bit are:

0 Access flag faults are enabled.

1 Access flag faults are not enabled.

If enabled, Access flag faults are reported by SMMU_CBn_FSR.

If AFFD==0, AP[0]==0 in the translation table entry causes an Access flag fault, which SMMU_CBn_FSR reports.

If AFFD==1, hardware behaves as if AP[0]==1 regardless of the translation table entry value.

For more information about the *Access permission* (AP) bit, see the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition*.

AFE, bit[2] Access Flag Enable. This bit is UNK/SBOP.

Stage 2 translation supports only the Long-descriptor translation table format, meaning that the descriptors support only the simplified model for access permissions. ARM recommends that software treats this bit as UNK/SBOP.

TRE, bit[1] TEX Remap Enable. This bit is UNK/SBOP.

Stage 2 translation supports only the Long-descriptor translation table format, meaning that the descriptors support only the simplified model for access permissions. ARM recommends that software treats this bit as UNK/SBOP.

M, bit[0] MMU Enable, 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.

16.3.3 SMMU_CB*n*_TTBCR, Translation Table Base Control Register

The SMMU_CB*n*_TTBCR characteristics are:

Purpose Provides additional configuration for the stage 2 translation process.

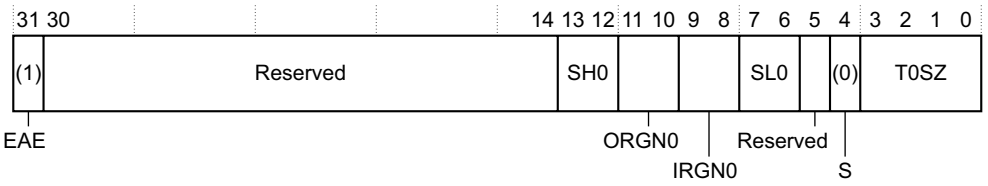
Usage constraints No usage constraints apply.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 2 translation. See:

- SMMU_IDR0.S2TS
- SMMU_IDR0.NTS
- *Translation context bank on page 2-27*.

Attributes A 32-bit RW register.

The SMMU_CB*n*_TTBCR bit assignments are:



EAE(1), bit[31]

Extended Address Enable.

For a Stage 2 translation context entry, this field always reads as the value 1. Writes are ignored.

A value of 1 means use the translation system defined in the LPAE.

Bits[30:14] Reserved.

SH0, bits[13:12]

Shareability attributes for the memory associated with the translation table walks using SMMU_CBn_TTBR0.

ORGN0, bits[11:10]

Outer cacheability attributes for the memory associated with the translation table walks using SMMU_CBn_TTBR0.

IRGN0, bits[9:8]

Inner cacheability attributes for the memory associated with the translation table walks using SMMU_CBn_TTBR0.

SL0, bits[7:6] Start Level for the SMMU_CBn_TTBR0 addressed region. The encoding of this field is:

0 Level 2.

1 Level 1.

2 UNPREDICTABLE.

3 UNPREDICTABLE.

Bit[5] Reserved.

S(0), bit[4] This bit must be programmed to T0SZ[3]. Otherwise, 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.

T0SZ, bits[3:0] The Size offset of the SMMU_CBn_TTBR0 addressed region, encoded as a 4-bit signed number giving the size of the region as $2^{32-T0SZ}$.

16.3.4 SMMU_CB*n*_TTBR0, Translation Table Base Register

The SMMU_CB*n*_TTBR0 characteristics are:

Purpose Holds the base address of translation table 0.

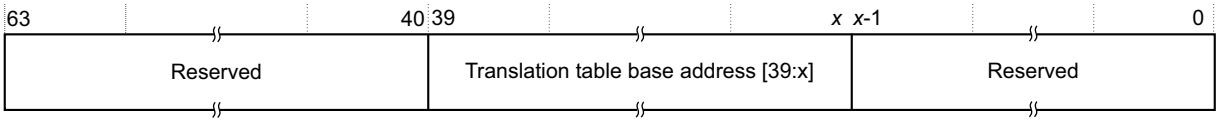
Usage constraints For a Stage 2 Translation context bank, only the Long-descriptor translation table format is supported.

Configurations It is IMPLEMENTATION DEFINED whether the system implements stage 2 translation. See:

- SMMU_IDR0.S2TS
- SMMU_IDR0.NTS
- *Translation context bank on page 2-27*.

Attributes A 64-bit RW register.

The SMMU_CB*n*_TTBR0 bit assignments are:



Bits[63:40] Reserved.

Translation table base address [39:*x*], bits[39:*x*]

Translation table base address, bits[39:*x*].

The value *x* is determined by SMMU_CBn_TTBCR.T0SZ using the following formula:

```
T0Size = SInt(SMMU_CBn_TTBCR.T0SZ);
if SMMU_CBn_TTBCR.SL0 == 0 then
   x = 14 - T0Size;
else
   x = 5 - T0Size;
```

Bits[*x*-1:0] Reserved.

Glossary

Abort An exception caused by an illegal memory access.

Banked register
A register that has multiple instances, with the instance that is in use depending on the processor mode, security state, or other processor state.

Byte An 8-bit data item.

Exception Handles an event. For example, an exception could handle an external interrupt or an undefined instruction.

External abort An abort that is generated by the external memory system.

Halfword-aligned
Means that the address is divisible by 2.

IMP An abbreviation used in diagrams to indicate that one or more bits have IMPLEMENTATION DEFINED behavior.

IMPLEMENTATION DEFINED
Means that the behavior is not architecturally defined, but must be defined and documented by individual implementations.

Intermediate Physical Address (IPA)
In an implementation of virtualization, the address to which a Guest OS maps a VA.

See also Physical address (PA), Virtual address (VA).

IPA *See* Intermediate Physical Address (IPA).

Little-endian memory
Means that:

- a byte or halfword at a word-aligned address is the least significant byte or halfword in the word at that address
- a byte at a halfword-aligned address is the least significant byte in the halfword at that address.

Memory coherency
Is the problem of ensuring that when a memory location is read, either by a data read or an instruction fetch, the value actually obtained is always the value that was most recently written to the location. This can be difficult when there are multiple possible physical locations, such as main memory and at least one of a write buffer and one or more levels of cache.

Memory Management Unit (MMU)
Provides detailed control of the part of a memory system that provides a single stage of address translation. Most of the control is provided using translation tables that are held in memory, and define the attributes of different regions of the physical memory map.

MMU *See* Memory Management Unit (MMU).

Nested translation Stage 2 translation.

Offset addressing
Means that the memory address is formed by adding or subtracting an offset to or from the base register value.

PA *See* Physical address (PA).

Physical address (PA)
Identifies a main memory location.

See also Intermediate Physical Address (IPA), Virtual address (VA).

RAZ *See* Read-As-Zero (RAZ).

RAZ/SBZP Read-As-Zero, Should-Be-Zero-or-Preserved on writes.

In any implementation, the bit must read as 0, or all 0s for a bit field, and writes to the field must be ignored.

Software can rely on the bit reading as 0, or all 0s for a bit field, but must use an SBZP policy to write to the field.

RAZ/WI Read-As-Zero, Writes Ignored.

In any implementation, the bit must read as 0, or all 0s for a bit field, and writes to the field must be ignored.

Software can rely on the bit reading as 0, or all 0s for a bit field, and on writes being ignored.

Read-allocate cache
Is a cache in which a cache miss on reading data causes a cache line to be allocated into the cache.

Read-As-Zero (RAZ) In any implementation, the bit must read as 0, or all 0s for a bit field.

Reserved Unless otherwise stated, bit positions described as reserved are UNK/SBZP.

SBO *See* Should-Be-One (SBO).

SBOP *See* Should-Be-One-or-Preserved (SBOP).

SBZ *See* Should-Be-Zero (SBZ).

SBZP *See* Should-Be-Zero-or-Preserved (SBZP).

Security hole Is a mechanism that bypasses system protection.

Should-Be-One (SBO) Should be written as 1, or all 1s for a bit field, by software. Values other than 1 produce UNPREDICTABLE results.

Should-Be-One-or-Preserved (SBOP)
Must be written as 1, or all 1s for a bit field, by software if the value is being written without having been previously read, or if the register has not been initialized. Where the register was previously read on the same processor, since the processor was last reset, the value in the field should be preserved by writing the value that was previously read.

Hardware must ignore writes to these fields.

If a value is written to the field that is neither 1 (or all 1s for a bit field), nor a value previously read for the same field on the same processor, the result is UNPREDICTABLE.

Should-Be-Zero (SBZ)
Should be written as 0, or all 0s for a bit field, by software. Values other than 0 produce UNPREDICTABLE results.

Should-Be-Zero-or-Preserved (SBZP)
Must be written as 0, or all 0s for a bit field, by software if the value is being written without having been previously read, or if the register has not been initialized. If the register was previously read, by the same processor that is writing to the register, since the processor was last reset, the value in the field should be preserved by writing the value that was previously read from the field by that processor.

Hardware must ignore writes to these fields.

If a value is written to the field that is neither 0 (or all 0s for a bit field), nor a value previously read for the same field on the same processor, software must expect an UNPREDICTABLE result.

TLB
See Translation Lookaside Buffer (TLB).

Transaction security state
Whether a transaction originates from a Secure or a Non-secure device. A transaction from a Secure device can make both Secure and Non-secure MMU accesses. A transaction from a Non-secure device can only make Non-secure accesses.

Translation Lookaside Buffer (TLB)
Is a memory structure containing the results of translation table walks. They help to reduce the average cost of a memory access. Usually, there is a TLB for each memory interface of the ARM implementation.

Translation table
A table held in memory that defines the properties of memory areas of various sizes from 1KB to 1MB.

Translation table walk
The process of doing a full translation table lookup. It is performed automatically by hardware.

UNDEFINED
Indicates an instruction that generates an Undefined Instruction exception.

For more information, see the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition*.

UNKNOWN
An UNKNOWN value does not contain valid data, and can vary from moment to moment, instruction to instruction, and implementation to implementation. An UNKNOWN value must not be a security hole. UNKNOWN values must not be documented or promoted as having a defined value or effect.

UNK/SBOP
UNKNOWN on reads, Should-Be-One-or-Preserved on writes.

In any implementation, the bit must read as 1, or all 1s for a bit field, and writes to the field must be ignored.

Software must not rely on the bit reading as 1, or all 1s for a bit field, and must use an SBOP policy to write to the field.

UNK/SBZP
In any implementation, the bit must read as 0, or all 0s for a bit field, and writes to the field must be ignored.

Software must not rely on the bit reading as 0, or all 0s for a bit field, and except for writing back to the register must treat the value as if it is UNKNOWN. Software must use an SBZP policy to write to the field.

See also UNK, Should-Be-Zero-or-Preserved (SBZP).

UNK
An abbreviation indicating that software must treat a field as containing an UNKNOWN value.

In any implementation, the bit must read as 0, or all 0s for a bit field. Software must not rely on the field reading as zero.

See also UNKNOWN.

UNPREDICTABLE
Means the behavior cannot be relied on. UNPREDICTABLE behavior must not perform any function that cannot be performed at the current or lower level of privilege using instructions that are not UNPREDICTABLE.

UNPREDICTABLE behavior must not be documented or promoted as having a defined effect.

An instruction that is UNPREDICTABLE can be implemented as UNDEFINED.

In an implementation that includes the Virtualization Extensions, execution in a Non-secure PL1 or PL0 mode of an instruction that is UNPREDICTABLE can be implemented as generating a Hyp Trap exception, provided that at least one instruction that is not UNPREDICTABLE causes a Hyp Trap exception.

VA *See* Virtual address (VA).

Virtual address (VA) An address generated by an ARM processor. For a PMSA implementation, the virtual address is identical to the physical address.

See also Intermediate Physical Address (IPA), Physical address (PA).

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

Word-aligned Means that the address is divisible by 4.

Write-Allocate cache
A cache in which a cache miss on storing data causes a cache line to be allocated into the cache.

Write-Back cache
A cache in which when a cache hit occurs on a store access, the data is only written to the cache. Data in the cache can therefore be more up-to-date than data in main memory. Any such data is written back to main memory when the cache line is cleaned or re-allocated. Another common term for a Write-Back cache is a *copy-back cache*.

Write-Through cache
A cache in which when a cache hit occurs on a store access, the data is written both to the cache and to main memory. This is normally done using a write buffer, to avoid slowing down the processor.