Arm® Corstone™ SSE-700 Subsystem

Revision: r1p0

Technical Reference Manual

arm

101418_0100_07_en

Arm® Corstone™ SSE-700 Subsystem

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-04	11 September 2019	Confidential	First release for r0p0 Beta
0000-05	13 December 2019	Confidential	First release for r0p0 LAC
0000-06	13 February 2020	Confidential	Second release for r0p0 LAC
0100-07	16 July 2020	Non-Confidential	First release for r1p0 EAC

Non-Confidential Proprietary Notice

(LES-PRE-20349)

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

www.arm.com

Contents
Arm® Corstone™ SSE-700 Subsystem Technical Reference Manual

Appendix D Revisions

Preface

This preface introduces the *Arm® Corstone™ SSE-700 Subsystem Technical Reference Manual*.

It contains the following:

About this book

This book is for the Arm® SSE-700 subsystem.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

r*m* Identifies the major revision of the product, for example, r1.

p*n* Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This book is written for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the SSE-700.

Using this book

This book is organized into the following chapters:

Chapter 1 Overview

This chapter introduces the Arm Corstone SSE-700 subsystem (SSE-700).

Chapter 2 Configuration options

This chapter describes the SSE-700 subsystem configurable options.

Chapter 3 Interfaces

This chapter describes the SSE-700 interfaces.

Chapter 4 Clocks

This chapter describes the primary and internal clocks within the SSE-700.

Chapter 5 Power

This chapter describes the voltage and power domains within the SSE-700.

Chapter 6 Reset

This chapter describes the reset requests, inputs, and outputs of the SSE-700.

Chapter 7 Debug

This chapter describes the SSE-700 debug infrastructure.

Chapter 8 Interconnect

This chapter describes the Host System Interconnect, which connects masters and slaves of the Host System.

Chapter 9 Host System peripherals

This chapter describes the peripherals, which are part of the SSE-700 Host System.

Chapter 10 Boot

This chapter describes the boot requirements and flow of the SSE-700.

Chapter 11 Programmers model

This chapter describes the SSE-700 memory and interrupt maps, and provides detailed information for all programmable registers.

Chapter 12 Software sequences

This chapter describes the software sequences that control functions within the SSE-700.

Appendix A Message Handling Unit

This appendix describes the *Message Handling Unit* (MHU) component of the SSE-700.

Appendix B Interrupt Router

This appendix describes the Interrupt Router.

Appendix C Firewall

This appendix describes the Firewall.

Appendix D Revisions

This appendix describes changes between released issues of this book.

Glossary

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

See the *Arm® Glossary* for more information.

Typographic conventions

italic

Introduces special terminology, denotes cross-references, and citations.

bold

Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace`

Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

`mono`space

Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic`

Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold`

Denotes language keywords when used outside example code.

`<and>`

Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

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

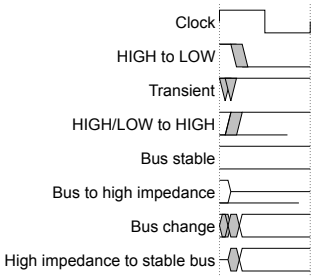

Figure 1 Key to timing diagram conventions

Signals

The signal conventions are:

Signal level

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

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

Lowercase n

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

Additional reading

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

Arm publications

- *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual* (101870)
- *Advanced Communications Channel Architecture Specification* (IHI 0076)
- *AMBA® Specification* (IHI 0011)
- *AMBA® APB Protocol Specification Version 2.0* (IHI 0024)
- *AMBA® 4 ATB Protocol Specification ATBv1.0 and ATBv1.1* (IHI 0032)
- *Arm® AMBA® 5 AHB Protocol Specification* (IHI 0033)
- *AMBA® AXI and ACE Protocol Specification* (IHI 0022)
- *AMBA® 4 AXI4-Stream Protocol Specification* (IHI 0051)
- *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces* (IHI 0068)
- *CoreLink™ GIC-400 Generic Interrupt Controller Technical Reference Manual* (DDI 0471)
- *Arm® CoreLink™ NIC-450 Network Interconnect Technical Overview* (100459)
- *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual* (101150)
- *Arm® CoreSight™ SDC-600 Secure Debug Channel Technical Reference Manual* (101130)
- *Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual* (100806)
- *Arm® CoreSight™ STM-500 System Trace Macrocell Technical Reference Manual* (DDI 0528)
- *Arm® CoreSight™ Architecture Specification v3.0* (IHI 0029)
- *Cortex®-M0+ Technical Reference Manual* (DDI 0484)
- *Arm® Cortex®-M System Design Kit Technical Reference Manual* (DDI 0479)
- *Arm® Cortex®-A32 Processor Technical Reference Manual* (100241)
- *Arm® Debug Interface Architecture Specification ADIv6.0* (IHI 0074)
- *Arm® Generic Interrupt Controller Architecture Specification, architecture version 2.0* (IHI 0048)
- *Arm® Power Policy Unit Architecture Specification, version 1.1* (DEN 0051)
- *Arm® Server Base System Architecture 5.0* (DEN 0029)
- *Arm®v6-M Architecture Reference Manual* (DDI 0419)
- *Arm®v7-M Architecture Reference Manual* (DDI 0403)
- *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* (DDI 0487)
- *PrimeCell UART (PL011) Technical Reference Manual* (DDI 0183)

The following confidential books are only available to licensees or require registration with Arm:

- *Arm® Corstone™ SSE-700 Subsystem Configuration and Integration Manual* (101419)
- *Arm® Corstone™ SSE-700 Subsystem Release Note* (PJDOC-1779577084-1555)
- *Arm® CoreSight™ System-on-Chip SoC-600 Configuration and Integration Manual* (100807)
- *Arm® Cortex®-A32 Processor Configuration and Sign-off Guide* (100244)
- *Arm® Cortex®-A32 Processor Integration Manual* (100245)
- *Cortex®-M0+ Integration and Implementation Manual* (DIT 0032)
- *Arm® Debug Interface Architecture Specification ADIv6.0* (IHI 0074)
- *Arm® Power Control System Architecture Specification* (DEN 0050)

Other publications

Feedback

Feedback on this product

If you have any comments or suggestions about this product, send an email to *support-subsystem-iot@arm.com* and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to *errata@arm.com*. Give:

- The title *Arm Corstone SSE-700 Subsystem Technical Reference Manual*.
- The number 101418_0100_07_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

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

Chapter 1
Overview

This chapter introduces the Arm Corstone SSE-700 subsystem (SSE-700).

It contains the following sections:

1.1 About SSE-700

The Arm Corstone SSE-700 is a flexible subsystem designed to provide a secure solution for rich *Internet of Things* (IoT) applications based on Arm Cortex-A32, Cortex-M, or other masters that are present in External Systems.

SSE-700 is designed to cover a range of *Power, Performance, and Area* (PPA) applications, and enable extension for use-case specific applications, for example, sensors, cloud connectivity, and edge compute.

Connected embedded devices are exposed to many different security threats. SSE-700 implements the reference subsystem for a SoC that targets secure, rich IoT applications. Built-in security is one of its key features.

SSE-700 consists of:

- A reference subsystem
- An example integration layer
- An example Cortex-A32 and CoreLink GIC-400 instantiation
- A set of documentation

To create a SoC, the SSE-700 subsystem must be extended. A complete system typically contains the following components:

Compute subsystem
: In SSE-700, the compute subsystem consists of one to four Cortex-A32 processor cores and associated bus, debug, controller, peripherals, and interface logic.

Memory and peripherals
: An extended SoC requires extra memory and peripheral components beyond the minimum subsystem components. Flash memory, for example, is not provided with the SSE-700, but can be added through implemented interfaces.

Sensors and actuators
: The reference design can be extended by adding sensors or actuator logic, such as temperature input or motor control output.

Software development environment
: Arm provides a complete software development environment, which includes the Arm Mbed™ *Operating System* (OS) and Linux, Mbed or GNU (GCC) compilers and debuggers, and firmware.
Custom peripherals typically require corresponding third-party firmware that can be integrated into the software stack.

1.2 Features of SSE-700

SSE-700 provides the minimum set of features for a SoC.

You can configure some aspects of the SSE-700 design to meet the specific requirements of your SoC and intended application. For example, you can configure the number of shared interrupt inputs or Host System Firewalls.

SSE-700 standardizes the following product features:

- Memory and interrupt maps of the Host System and Secure Enclave system
- Hardware address compartmentalization
- Boot and security, including a hardware isolated *Root of Trust* (RoT)
- Communication between different parts of the SoC
- Timer and watchdog infrastructure
- Debug requirements
- Power, clock, and reset control
- Requirements for adding new or existing External Systems for use-case specific applications
- Requirements for adding use-case specific masters and peripherals to the Host System

For more information on the topology of SSE-700 and the different types of systems, such as Host System, Secure Enclave system, and External System, see *1.3 SSE-700 topology* on page 1-17.

In this document, the term *integrator* refers to the person who integrates the SSE-700 design into a SoC. The integrator implements the rest of the SoC following the requirements made by SSE-700.

The SSE-700 includes the following key components:

- A Cortex-A processor cluster and *Generic Interrupt Controller* (GIC) socket to support up to four Cortex-A32 cores and a CoreLink GIC-400
- Secure Enclave socket, based on a Cortex-M0+ processor with dedicated SRAM, ROM, and peripherals, such as timers and watchdogs, into which a Crypto Accelerator can be integrated
- Two External System Harnesses support integration of External Systems into SSE-700
- *Message Handling Units* (MHUs) for communication between the different systems in the SSE-700. An MHU provides a unidirectional communication channel, therefore MHUs are provided in pairs to allow for bidirectional communication:
  - Two pairs of MHUs for communication between Cortex-A32 and Secure Enclave
  - Two pairs of MHUs for communication between Cortex-A32 and each External System
  - Two pairs of MHUs for communication between Secure Enclave and each External System
- Interrupt Router to handle routing interrupts from shared peripherals to the interrupt controller of a specific system
- Firewall to provide:
  - Hardware compartmentalization of the Host System address space
  - Translation between the address space of the Secure Enclave system and the External System into the address space of the Host System
- Common debug infrastructure supporting single and multi-system debug, by self-hosted and external debug agents
- Certificate-based debug authentication is supported by SDC-600
- Advanced hardware autonomous power control design
- Shared peripherals, such as counters, timers, and watchdog, that can be used by any system in the SSE-700

The integrator can tailor the final implementation to the target use-cases by using:

- The supported configuration options, see *Chapter 2 Configuration options* on page 2-22
- The sockets for the Host CPU, Host GIC, and External Systems
- The expansion interfaces of SSE-700

1.3 SSE-700 topology

The SSE-700 design consists of several components:

- A Host System
- A Secure Enclave system
- Two External System Harnesses
- An Integration Layer that demonstrates how the SSE-700 design can be extended to create a SoC

Figure 1-1 SSE-700 topology on page 1-17 shows the topology of the SSE-700. An SSE-700-based SoC is formed of three different types of system:

- Secure Enclave: green box
- External Systems: red box
- Host System: the rest of the SSE-700 subsystem

The Secure Enclave socket and External System Harnesses are connected, with all other master and peripherals added by the integrator, to the interconnect inside the Host System. Each system provides different functionality to the overall SoC. They operate as self-contained entities, independent of the other systems in the SoC.

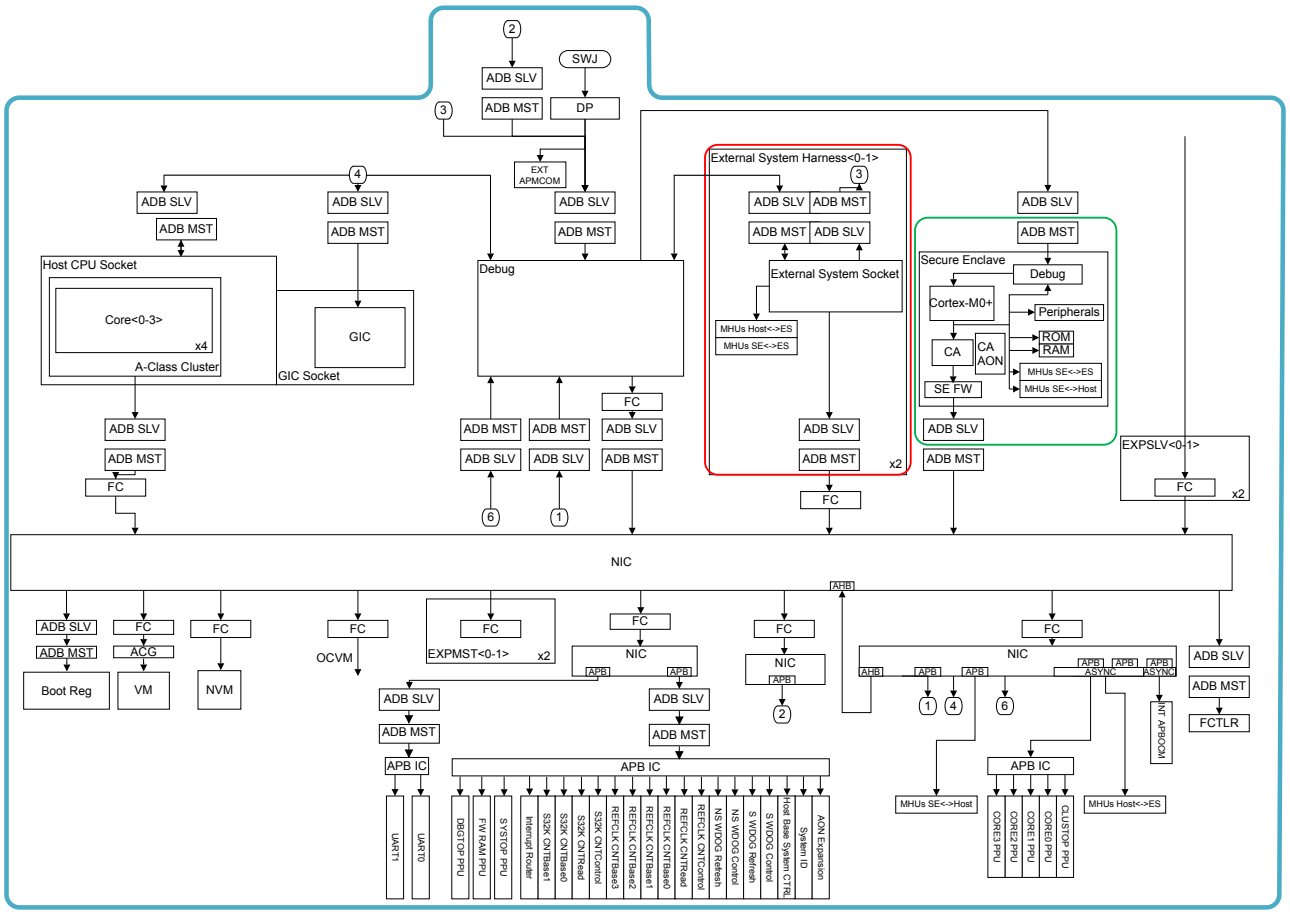

Figure 1-1 SSE-700 topology

Note

In *Figure 1-1 SSE-700 topology* on page 1-17, the Secure Enclave and External Systems are not part of the Host System.

This section contains the following subsections:

1.3.1 Host System

The Host System is based on an Arm A-profile processor with standard peripherals that enable booting a Rich OS.

The Host System provides access to resources that can be shared with the Secure Enclave or External System, such as volatile and non-volatile memory. The Host System includes a Firewall that provides hardware compartmentalization of the Host System address space. This compartmentalization enables software to create sandboxes between the different systems, and to allow only specific regions to be used for communication between the systems.

The Host System provides:

- Common timestamps for use with Generic Timer and Watchdog as defined in the *Arm Architecture Reference Manual Armv8 for Armv8-A architecture profile*
- UARTs
- MHUs for communication with the Secure Enclave and External Systems
- Routing of interrupts, from shared peripherals, to the interrupt controller associated with the software entity controlling the peripheral
- Shared secured debug infrastructure with support for:
  - Trace
  - Self-hosted and External debug of all systems

    Note

    There is no self-hosted debug on the Secure Enclave because Cortex-M0+ does not support it.

  - Functional I/O debug
  - Multi-system debug

The Host System defines a Host CPU and *Generic Interrupt Controller* (GIC) Socket. These are the interfaces that allow integration of the following supported A-profile processors or GICs:

- Cortex-A32
- CoreLink GIC-400

1.3.2 Secure Enclave

The Secure Enclave system provides *Root of Trust* (RoT) and cryptographic functions.

The Secure Enclave system is based on a Cortex-M0+ processor core and associated peripherals, such as timers and watchdogs. The Secure Enclave requires that the integrator adds a Crypto Accelerator. The algorithms that the Crypto Accelerator supports are IMPLEMENTATION DEFINED. The Crypto Accelerator is not part of the SSE-700. Partners can integrate their logic into the Secure Enclave socket interface to provide a hardware-based cryptography function.

Hardware isolates the software running on the Secure Enclave system. This reduces the complexity in security reviews. Communication with the Secure Enclave system is achieved using MHUs.

For more information about the Secure Enclave system, see the *Arm Corstone SEE-700 Secure Enclave Technical Reference Manual*.

1.3.3 External Systems

An External System provides SoC use-case specific functionality.

One example of an External System is a pre-existing MCU, which provides connectivity or sensor monitoring functionality.

The SSE-700 design does not specify what can be included as part of an External System. However, the External System must meet several requirements to integrate into a SoC. To support this, SSE-700 has two External System Harnesses for the integration of the External Systems. Each External System is considered an independent system and can contain any processors, peripherals, and masters required to meet the use cases of the External System.

The External System Harness interfaces enable the External System to:

- Access the address space of the Host System
- Communicate with the Host and Secure Enclave systems
- Connect to the shared SoC debug infrastructure

For more information on the interfaces of the External System Harness, see *3.3 External System Harness interface* on page 3-38.

1.4 Compliance

The SSE-700 complies with, or includes components that comply with, the following specifications.

- Arm Architecture
- Debug
- Interrupt controller architecture
- *Power Policy Unit* (PPU) architecture
- *Advanced Microcontroller Bus Architecture* (AMBA)

This *Technical Reference Manual* (TRM) complements the TRMs for included components, Architecture Reference Manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources.

This section contains the following subsections:

1.4.1 Arm architecture

The Cortex-A32 processor, supported by SSE-700, implements the Armv8-A architecture, which executes the A32 and T32 instruction sets.

The Cortex-M0+ processor in the SSE-700 implements the Armv6-M architecture profile.

For more information, see the *Armv6-M Architecture Reference Manual* and *Arm Architecture Reference Manual Armv8 for Armv8-A architecture profile*.

1.4.2 Debug

The SSE-700 implements Arm CoreSight SoC-600 and complies with the following specifications:

- *Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual*
- *Arm® Debug Interface Architecture Specification ADIv6.0*
- *Arm® CoreSight™ Architecture Specification v3.0*

1.4.3 Interrupt controller architecture

The GIC-400 interrupt controller supported by the SSE-700 complies with the following specification:

Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 2.0

1.4.4 Power Policy Unit architecture

The CoreLink PCK-600 power management infrastructure in the SSE-700 complies with the following specifications:

- *Arm® Power Policy Unit Architecture Specification, version 1.1*
- *Arm® Power Control System Architecture Specification*
- *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*

1.4.5 Advanced Microcontroller Bus Architecture

The SSE-700 complies with the:

- *AMBA® AXI and ACE Protocol Specification*
- *Arm® AMBA® 5 AHB Protocol Specification*
- *Arm® AMBA® APB Protocol Specification*
- *AMBA® 4 ATB Protocol Specification ATBv1.0 and ATBv1.1*
- *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*

1.5 Product revisions

This section describes the differences in functionality between product revisions.

r0p0
BET - LAC
- DC-600 REMRR support removed
- Host CPU error read response data set to `0xB773_B773`
- Host Base System Control, Part ID changed to `0x78`
- Secure Enclave Base System Control Register Part ID changed to `0x77`
- Firewall technical updates

r0p0-r1p0

ID register value changed to reflect product revision status:
- System ID Implementer Identification Register 0x7491043B
- System ID PID2 Register 0x0000001B
- Firewall Controller Implementer Identification Register 0x0750143B
- Firewall Controller PID3 Register 0x00000010

Various engineering errata fixes.

Chapter 2
Configuration options

This chapter describes the SSE-700 subsystem configurable options.

It contains the following sections:

2.1 SSE-700 configuration

The following table shows the SSE-700 configuration options.

Table 2-1 SSE-700 configuration

Configuration option	Legal values	Description
NUM_EXP_SHD_INT	0-395	Number of supported shared interrupt inputs
SI{x}_ICI_DST	0x00 – 0x3F	Allowed destinations, which shared interrupt x can be routed to
SI{x}_DEF_ICI	0x00 – 0x3F	Default routing, for shared interrupt x
NUM_ACLK_QCH	0-8	Number of expansion **ACLK** Q-Channels, in addition to those for the Expansion Slave and Master, and OCVM interfaces
NUM_DBGCLK_QCH	1-8	Number of expansion **DBGCLK** Q-Channels

2.2 Host System configuration

The following table shows the Host System configuration options.

Table 2-2 Host System configuration

Configuration option	Legal values	Description
HOST_CPU_NUM_CORES	1-4	Selects the number of Host CPU cores
HOST_EXP_ROM_ENTRY	0x0000_0000, 0x0100_0007 - 0x01FF_F007	CoreSight ROM table entry in the Host ROM table to point to expansion debug logic. This can be an entry to another CoreSight ROM table or a single debug component: 0x0000_0000: No additional debug logic is added 0x0100_0007 - 0x01FF_F007: Base address of the additional debug component or ROM table added on HOSTDBGEXP interface

SSE-700 supports up to four Cortex-A32 cores.

The only *Generic Interrupt Controller* (GIC) supported by the SSE-700 is GIC-400. All other GICs are not supported.

2.3 External System configuration

The following table shows the External System configuration options.

Table 2-3 External System configuration

Configuration option	Legal values	Description
EXT_SYS0_ROM_ENTRY	0x0000_0000, 0x001D_0017 -0x002C_F017	CoreSight ROM Table entry in the EXTDBG ROM to point to the External System's debug logic. This can be an entry to a CoreSight ROM table or a single debug component: 0x0000_0000: No debug logic is provided by the External System 0 0x001D_0017 -0x002C_F017: Base address of the additional debug component or ROM table added by the External System using the EXTSYS0DBG interface
EXT_SYS1_ROM_ENTRY	0x0000_0000, 0x002D_0027 - 0x003C_F027	CoreSight ROM Table entry in the EXTDBG ROM to point to the External System's debug logic. This can be an entry to a CoreSight ROM table or a single debug component: 0x0000_0000: No debug logic is provided by the External System 1 0x002D_0027 - 0x003C_F027: Base address of the additional debug component or ROM table added by the External System using the EXTSYS1DBG interface

2.4 Host System Firewall configuration

The following table shows the Host System Firewall configuration options.

Table 2-4 Host System Firewall configuration

Configuration option	Legal values	Description
FIREWALL_F0_CFG_SSE700_AONPERIPH_FC_RGN{x}_BASE_ADDR	0x1A60_0000 - 0x1A6F_FFFF	AONPERIPH Firewall Component region x base address, where x is 24-39
FIREWALL_F0_CFG_SSE700_AONPERIPH_FC_RGN{x}_SIZE	0x00, 0x0C-0x14	AONPERIPH Firewall Component region x size, where x is 24-39
XNVM_NUM_RGN	16, 32, 48, 64	Number of regions for the XNVM Firewall Component
XNVM_RSE_LVL	0, 1	Level of RSE implemented by the XNVM Firewall Component
CVM_NUM_RGN	16, 32, 48, 64	Number of regions for the CVM Firewall Component
CVM_RSE_LVL	0, 1	Level of RSE implemented by the CVM Firewall Component
DBG_NUM_RGN	4, 8	Number of regions for the DBG Firewall Component
EXTSYS0_NUM_RGN	8, 16	Number of regions for the EXTSYS0 Firewall Component
EXTSYS1_NUM_RGN	8, 16	Number of regions for the EXTSYS1 Firewall Component
EXPSLV0_NUM_RGN	8, 16, 32	Number of regions for the EXPSLV0 Firewall Component
EXPSLV1_NUM_RGN	8, 16, 32	Number of regions for the EXPSLV1 Firewall Component
EXPMST0_PE_LVL	1, 2	Level of PE implemented by the EXPMST0 Firewall Component.
EXPMST0_RSE_LVL	0, 1	Level of RSE implemented by the EXPMST0 Firewall Component. This value must be 0 when EXPMST0_PE_LVL is 1.
EXPMST0_NUM_RGN	When PE.1 implemented: 1-64 When PE.2 implemented: 8, 16, 32	Number of regions for EXPMST0 Firewall Component
FIREWALL_F0_CFG_SSE700_EXPMST0_FC_RGN{x}_BASE_ADDR	0x4000_0000 - 0x5FFF_FFFF	EXPMST0 region x base address, where x is between 0 and EXPMST0_NUM_RGN-1. This value is ignored when EXPMST0_PE_LVL is 2.
FIREWALL_F0_CFG_SSE700_EXPMST0_FC_RGN{x}_SIZE	0, 0x0C-0x1D	EXPMST0 region x is defined as a power of 2 or multiple of the MNRS, where x is between 0 and EXPMST0_NUM_RGN-1. This value is ignored when EXPMST0_PE_LVL is 2.
EXPMST0_MXRS	29	MXRS value for EXPMST0 Firewall Component
EXPMST1_MXRS	29	MXRS value for EXPMST1 Firewall Component
EXPMST1_NUM_RGN	8, 16, 32	Number of regions for EXPMST1 Firewall Component
EXPMST1_RSE_LVL	0, 1	Level of RSE implemented by the EXPMST1 Firewall Component
EXPMST1_PE_LVL	2	Level of PE implemented by the EXPMST1 Firewall Component
OCVM_NUM_RGN	16, 32, 64	Number of regions for the OCVM Firewall Component

Table 2-4 Host System Firewall configuration (continued)

Configuration option	Legal values	Description
OCVM_RSE_LVL	0, 1	Level of RSE implemented by the OCVM Firewall Component
FC_NUM_MST_ID	7-231	Number of StreamIDs which have a unique value set by FIREWALL_F0_CFG_GLOBAL_ SSE700_FC_MST_ID{x}_VAL and FIREWALL_F0_CFG_GLOBAL_SSE700_ ERR_RESP_PER_MST_ID{x}_VAL configuration options. This value is a globally defined value for the Host system Firewall network.
FC_ERR_RESP_DEF[a]	0x0000_0000 - 0xFFFF_FFFF	Default data value for error response when StreamID value does not match the pre-configured values, and provides data value for error response when SINGLE_MST is set to 1 for a Firewall
FIREWALL_F0_CFG_GLOBAL _SSE700_FC_MST_ID{x}_VAL	0x20 - 0xFF	Array of parameters containing the list of supported StreamID values. The size of array is defined by FC_NUM_MST_ID. x must be between 7 and FC_NUM_MST_ID-1. The configured value is StreamID of the issuing master. This parameter is not used for a Firewall component that has the parameter SINGLE_MST set to 1.
FIREWALL_F0_CFG_GLOBAL_ SSE700_ERR_RESP_PER_MST_ ID{x}_VAL	0x0000_0000 - 0xFFFF_FFFF	Unique error response data value to be returned as part of read response corresponding to the StreamID value defined in array. Where x must be between 7 and FC_NUM_MST_ID-1. This parameter is not used for a Firewall component that has the parameter SINGLE_MST set to 1. Instead the FC_ERR_RESP_DEF parameter is used to return appropriate read data on an error response.

[a] The different Firewall instances implemented in SSE-700 support different memory path widths, which could be 32-bit, 64-bit, or 128-bit. For 64-bit or 128-bit Firewalls, the defined 32-bit default data for error responses is duplicated to all 32-bit word lanes.

2.5 MHU configuration

The following table shows the MHU configuration option.

Table 2-5 MHU configuration

Configuration option	Legal values	Description
MHU_{x}_NUM_CH	1-32	Number of Channels supported on an MHU, where x is the MHU short name as listed in *9.5 MHU* on page 9-152

2.6 IP configuration

Some of the IPs used within the SSE-700 subsystem provide configuration options.

The supported IP configuration options are listed below:

CoreLink ADB-400
The size of the FIFOs of ADBs can be set to any value to support the required bandwidth.

CoreLink GIC-400
All configuration options are supported. The number of SPIs must be between 73-480.

Cortex-A32
The SSE-700 supports all allowed Cortex-A32 configurations, with the following conditions:

- *Accelerator Coherency Ports* (ACPs) cannot be included.
- Master interfaces must be in AXI4 mode.
- ETM must be included.
- GIC CPU interface must not be present.

PCK-600
The following configuration options of the *Power Policy Unit* (PPU) are supported:

- `DEV_PREQ_DLY`
- `PCSM_PREQ_DLY`
- `ISO_CLKEN_DLY_CFG`
- `CLKEN_RST_DLY_CFG`
- `RST_HWSTAT_DLY_CFG`
- `CLKEN_ISO_DLY_CFG`
- `ISO_RST_DLY_CFG`

The GIC-400 and Cortex-A32 detailed configurations are not described in the SSE-700 product documentation. For configuration information about the Cortex-A32 and GIC-400, see the respective IP documents listed in *Additional reading* on page 11.

Chapter 3
Interfaces

This chapter describes the SSE-700 interfaces.

It contains the following sections:

3.1 Interfaces overview

The SSE-700 interfaces are defined with associated properties, such as the address and data width. All interfaces are defined with a clock, power, and reset domain, except for the clock and reset interfaces.

For more information on the clock, power, and reset domains of an SSE-700, see sections *Chapter 4 Clocks* on page 4-58, *5.1 Power overview* on page 5-69, and *6.1 Reset overview* on page 6-92.

The following conventions are used:

- Inputs and outputs are for SSE-700.
- An AMBA interface described as a master interface, is an interface where SSE-700 is the master. It is connected to a slave interface of a component that is outside the SSE-700. For an AXI master interface, the **ARVALID** signal is an output and **ARREADY** is an input.
- An AMBA interface described as a slave interface, is an interface where SSE-700 is the slave. It must be connected to a master interface of a component external to SSE-700. For an AXI slave interface, the **ARVALID** signal is an input and **ARREADY** is an output.
- A Q-Channel interface described as a control interface, is an interface where SSE-700 is the device. It must be connected to a control interface. For a Q-Channel control interface, the **QREQn** signal is an input and **QACCEPTn**, **QDENY**, and **QACTIVE** are outputs.
- A Q-Channel interface described as a device interface, is an interface where SSE-700 is the controller. It must be connected to a device interface. For a Q-Channel device interface, the **QREQn** signal is an output and **QACCEPTn**, **QDENY** and **QACTIVE** are inputs.

Note

Unless stated otherwise, all Q-Channel interfaces are single Q-Channel instances.

3.2 Host CPU-GIC socket interfaces

This section describes the interfaces that integrate a Host CPU and Host GIC into the SSE-700.

This section contains the following subsections:

3.2.1 Clock interface

The Host CPU Clock interface, **HOSTCPUCLKOUT**, is driven by the **HOSTCPUCLK**.The GIC Clock interface, **GICCLKOUT**, is driven by the **GICCLK**.

3.2.2 Reset interface

The Host CPU Reset interface includes the following reset signals:

- **HOSTCPUWARMRESETn[HOST_CPU_NUM_CORES-1:0]**: Warm reset for the CORE{x} power domain
- **HOSTCPUPORESETn[HOST_CPU_NUM_CORES-1:0]**: Power on reset for the CORE{x} power domain
- **HOSTCLUSTOPWARMRESETn**: Warm reset for the CLUSTOP power domain
- **HOSTCLUSTOPPORESETn**: Power on reset for the CLUSTOP power domain

The GIC reset signal, **GICRESETn**, is driven by **CLUSTOPWARMRESETn**.

3.2.3 Host CPU Memory (HOSTCPUMEM) interface

The HOSTCPUMEM interface is an AXI5 slave interface to be connected to a Host CPU AXI master interface.

The HOSTCPUMEM interface has the following properties:

- AXI5 with the following properties set to True:
  - Wakeup_Signals
- 40-bit address

  ——— **Note** ———

  Accesses outside the first 4GB result in a DECERR being generated by SSE-700.

  ———————————

- 128-bit data
- **HOSTCPUCLKOUT**
- CLUSTOP power domain
- **HOSTCLUSTOPWARMRESETn**

3.2.4 GIC Master (GICM) interface

The GICM interface is to be connected to the AXI slave interface of the Host CPU GIC.

This interface allows configuration access to the registers of the Host CPU GIC. It has the following properties:

- AXI5 with the following properties set to True:
  - Wakeup_Signals
  - Untranslated_Transactions
- 32-bit address
- 32-bit data
- 2 user bits for AR and AW Channels:
  - Indicate which Host CPU core generated the transaction
- **GICCLKOUT**
- CLUSTOP power domain
- **CLUSTOPWARMRESETn**

3.2.5 Host CPU Debug APB (HOSTCPUDBG) interface

The HOSTCPUDBG interface is an APB4 master interface to be connected to the APB4 slave debug interface of the Host CPU.

The HOSTCPUDBG interface has the following properties:

- APB4, including a **PWAKEUP** output
- 32-bit address and data
- **HOSTCPUCLKOUT**
- CLUSTOP power domain
- **CLUSTOPPORESETn**

3.2.6 Host CPU Trace (HOSTCPUTRACE) interface

The HOSTCPUTRACE interface is an ATB4 slave interface to be connected to the ATB4 master interfaces, and to be integrated to the Host CPU trace sources.

The HOSTCPUTRACE interface has the following properties:

- ATB4
- 32-bit data
- **HOSTCPUCLKOUT**
- CLUSTOP power domain
- **CLUSTOPPORESETn**

3.2.7 Host CPU CTI Channel In (HOSTCPUCTICHIN) interface

The HOSTCPUCTICHIN interface is an input CTI channel interface. It connects the CTIs within the Host CPU cores to the CTI network provided by SSE-700.

The interface has the following properties:

- CTI Channel interface
- **HOSTCPUCLKOUT**
- CLUSTOP power domain
- **CLUSTOPPORESETn**

3.2.8 Host CPU CTI Channel Out (HOSTCPUCTICHOUT) interface

The HOSTCPUCTICHOUT interface is an output CTI channel interface. It connects the CTIs of the Host CPU to the CTI network provided by the SSE-700.

The interface has the following properties:

- CTI Channel interface
- **HOSTCPUCLKOUT**

- CLUSTOP power domain
- **CLUSTOPPORESETn**

3.2.9 Host CPU Debug Authentication (HOSTCPUDBGAUTH) interface

The HOSTCPUDBGAUTH interface has the standard Arm debug authentication signals for use by the Host CPU:

- **DBGEN**, **NIDEN**, **SPIDEN**, **SPNIDEN**
- Asynchronous
- CLUSTOP power domain
- **SEPORESETn**

3.2.10 Host CPU Power Control (HOSTCPUPWR) interface

The HOSTCPUPWR interface contains signals for power, clock, and reset control.

The signals in the interface are listed below. In some cases, the width of the signal depends on SSE-700 configuration options.

- CPU Q-Channels – Power Q-Channel for the Host CPU in CORE{0-3} power domain:
  - **CPUQREQn[HOST_CPU_NUM_CORES-1:0]**
  - **CPUQACCEPTn[HOST_CPU_NUM_CORES-1:0]**
  - **CPUQDENY[HOST_CPU_NUM_CORES-1:0]**
  - **CPUQACTIVE[HOST_CPU_NUM_CORES-1:0]**
- L2 Q-Channel – Power Q-Channel for the L2 cache RAM:
  - **L2QREQn**
  - **L2QACCPETn**
  - **L2QDENY**
  - **L2QACTIVE**
- CLUSTOP Ingress Q-Channel. (Device Q-Channel interface):
  - **CLUSTOPINGRESSQREQn**
  - **CLUSTOPINGRESSQACCEPTn**
  - **CLUSTOPINGRESSQDENY**
  - **CLUSTOPINGRESSQACTIVE**
- CLUSTOP Egress Q-Channel. (Device Q-Channel interface):
  - **CLUSTOPEGRESSQREQn**
  - **CLUSTOPEGRESSQACCEPTn**
  - **CLUSTOPEGRESSQDENY**
  - **CLUSTOPEGRESSQACTIVE**
- **SMPEN[HOST_CPU_NUM_CORES-1:0]**
- **STANDBYWFI[HOST_CPU_NUM_CORES-1:0]**
- **STANDBYWFIL2**
- **DBGPWRUPREQ[HOST_CPU_NUM_CORES-1:0]**
- **DBGNOPWRDWN[HOST_CPU_NUM_CORES-1:0]**
- **DBGPWRDUP[HOST_CPU_NUM_CORES-1:0]**
- **L2RSTDISABLE**
- **WARMRSTREQ[HOST_CPU_NUM_CORES-1:0]**
- **DBGRSTREQ[HOST_CPU_NUM_CORES-1:0]**
- **L2FLUSHREQ**
- **L2FLUSHDONE**
- **WAKEUPREQ[HOST_CPU_NUM_CORES-1:0]**

3.2.11 Host CPU Configuration (HOSTCPUCFG) interface

The HOSTCPUCFG interface contains configuration signals for the Host CPU.

The signals in the HOSTCPUCFG interface are:

- **CFGEND[HOST_CPU_NUM_CORES-1:0]**
- **CFGTE[HOST_CPU_NUM_CORES-1:0]**
- **VINITHI[HOST_CPU_NUM_CORES-1:0]**
- **CRYPTODISABLE**
- **CP15SDISABLE[HOST_CPU_NUM_CORES-1:0]**

3.2.12 Host System Generic Timestamp Gray (HOSTCNTVALUEG) interface

The Host System Generic Timestamp Gray interface provides a gray-encoded timestamp input.

It is required to be converted to a binary value and connected to **CNTVALUEB** of the Host CPU. This interface has the following properties:

- 64-bit gray-encoded timestamp
- **HOSTCPUCLKOUT**
- CLUSTOP power domain
- **HOSTCLUSTOPWARMRESETn**

3.2.13 GIC Configuration (GICCFG) interface

The GIC Configuration Interface is made up of the **CFGSDISABLE** signal which is driven by the HOST_SYS_LCTRL_ST.HOST_GIC_LOCK field.

This interface has the following properties:

- Asynchronous
- CLUSTOP power domain
- **GICRESETn**

3.2.14 GIC Interrupt (GICINT) interface

The GIC Interrupt interface provides the interrupt outputs, which must be connected to the Host System GIC, where the source is within SSE-700.

The GIC Interrupt interface provides the interrupt outputs where the source is within SSE-700. These must be connected to the Host System GIC. The interface is made up of the following signals:

- **GICINTDBGTOP**
- **GICINTMHUS**
- **GICINTMHUNS**
- **GICINTUART**
- **GICINTWDOGS**
- **GICINTWDOGNS**

The **GICINTDBGTOP** signal has the following bit assignments:

Table 3-1 GICINTDBGTOP bit assignments

Bit offset	Source	Power domain	Clock domain	GIC SPI number
0	Host STM Sync IRQ	DBGTOP	GICCLK	69
1	Host ETR Buffer IRQ	DBGTOP	DBGCLK	70
2	Host CATU IRQ	DBGTOP	DBGCLK	71
3	Host CTI Trigger Out 4	DBGTOP	GICCLK	72
4	Host CTI Trigger Out 5	DBGTOP	GICCLK	73

The **GICINTMHUS** signal has the following bit assignments:

Table 3-2 GICINTMHUS bit assignments

Bit offset	Source	Power domain	Clock domain	GIC SPI number	Notes
0	Host to Secure Enclave MHU0 Combined IRQ	SYSTOP	ACLK	41	-
1	Secure Enclave to Host MHU0 Combined IRQ	SYSTOP	ACLK	42	-
2	Host to EXTSYS0 MHU0 Combined IRQ	SYSTOP	ACLK	43	-
3	EXTSYS 0 to Host MHU0 Combined IRQ	SYSTOP	ACLK	44	
4	Host to EXTSYS 1 MHU0 Combined IRQ	SYSTOP	ACLK	45	-
5	EXTSYS 1 to Host MHU0 Combined IRQ	SYSTOP	ACLK	46	
6-9	-	-	ACLK	47-50	Present and driven LOW

The **GICINTMHUNS** signal has the following bit assignments:

Table 3-3 GICINTMHUNS bit assignments

Bit offset	Source	Power domain	Clock domain	GIC SPI number	Notes
0	Host to Secure Enclave MHU1 Combined IRQ	SYSTOP	ACLK	76	-
1	Secure Enclave to Host MHU1 Combined IRQ	SYSTOP	ACLK	77	-
2	Host to EXTSYS 0 MHU1 Combined IRQ	SYSTOP	ACLK	78	-
3	EXTSYS 0 to Host MHU1 Combined IRQ	SYSTOP	ACLK	79	
4	Host to EXTSYS 1 MHU1 Combined IRQ	SYSTOP	ACLK	80	-
5	EXTSYS 1 to Host MHU1 Combined IRQ	SYSTOP	ACLK	81	
6-9	-	-	ACLK	82-85	Present and driven LOW

The **GICINTUART** signal has the following bit assignments:

Table 3-4 GICINTUART bit assignments

Bit offset	Source	Power domain	Clock domain	GIC SPI number
0	UART0	AONTOP	UARTCLK	51
1	UART1	AONTOP	UARTCLK	52

The **GICINTWDOGS** signal has the following bit assignments:

Table 3-5 GICINTWDOGS bit assignments

Bit offset	Source	Power domain	Clock domain	GIC SPI number
0	Secure Watchdog WS0	AONTOP	REFCLK	32
1	Non-secure Watchdog WS1	AONTOP	REFCLK	33

The **GICINTWDOGNS** signal has the following bit assignment:

Table 3-6 GICINTWDOGNS bit assignment

Bit offset	Source	Power domain	Clock domain	GIC SPI number
0	Non-secure Watchdog WS0	AONTOP	REFCLK	64

3.2.15 GIC Shared Interrupt (GICSHDINT) interface

The GIC Shared Interrupt interface is driven by the ICI1 interface of the Interrupt Router.

The interface has the following properties:

- Asynchronous
- AONTOP power domain
- **AONTOPWARMRESWETn**
- Width is `NUM_EXP_SHD_INT` + 32

The bit assignment is as defined in *Table 9-1 Interrupt Router interface assignment* on page 9-150.

3.2.16 GIC Wakeup (GICWAKEUP) interface

This interface is the **GICWAKEUP** signal. This signal is an active-HIGH request for the Host System GIC to become operational.

The interface has the following properties:

- Asynchronous
- AONTOP power domain
- **AONTOPWARMRESETn**

3.3 External System Harness interface

This section describes the interfaces of the External System Harness.

All the interfaces are prefixed with `EXTSYS{x}`, where x is the number of the External System, between 0 and 1. Each External System has its own version of each interface, which is associated with the clock, reset, and power domain of that External System.

This section contains the following subsections:

3.3.1 External System {0-1} Clock interface

The clock inputs for this interface are used inside the External System Harness.

The External System {0-1} Clock interface is made up of the following clock inputs:

- **EXTSYS{0-1}DBGCLKS**
- **EXTSYS{0-1}DBGCLKM**
- **EXTSYS{0-1}ATCLK**
- **EXTSYS{0-1}CTICLK**
- **EXTSYS{0-1}ACLK**
- **EXTSYS{0-1}MHUCLK**

3.3.2 External System {0-1} Reset Output interface

This interface comprises control signals for the External System:

- **EXTSYS{0-1}PORESETn**: Power-on reset for all logic in the External System.

  ——— **Note** ———

  EXTSYS{0-1}PORESETn is a reset output from SSE-700, and a reset input to the External System.

  ———————————

- **EXTSYS{0-1}CPUWAIT**: Instruction execution halt for any master in the External System that executes instructions.

3.3.3 External System {0-1} Reset Input interface

The following reset signals for this interface are asynchronous assert and synchronous deassert.

They are used by the components and interfaces of the External System Harness:

- **EXTSYS{0-1}DBGPRESETSn**
- **EXTSYS{0-1}DBGPRESETMn**
- **EXTSYS{0-1}ATRESETn**
- **EXTSYS{0-1}CTIRESETn**
- **EXTSYS{0-1}ARESETn**
- **EXTSYS{0-1}MHURESETn**

3.3.4 External System {0-1} Memory (EXTSYS{0-1}MEM) interface

The External System {0-1} Memory interface is an AXI5 slave interface. It must be connected to an External System AXI master interface to provide access to the Host System.

The interface has the following properties:

- AXI5, with Wakeup_Signals set to True
- 32-bit address
- 64-bit data
- **EXTSYS{0-1}ACLK**
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}ARESETn**

3.3.5 External System {0-1} MHU (EXTSYS{0-1}MHU) interface

The External System {0-1} MHU access interface is an APB4 slave interface providing access to MHUs.

The interface has the following properties:

- APB4 including a **PWAKEUP** input
- 19-bit address.
- 32-bit data
- **EXTSYS{0-1}MHUCLK**
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}MHURESETn**

11.1.4 External System memory map on page 11-170 defines the memory map for this interface.

3.3.6 External System {0-1} MHU Interrupt (EXTSYS{0-1}MHUINT) interface

The External System {0-1} MHU Interrupt interface has 8 interrupt signals.

All signals use the following name format {x}EXTSYS{0-1}MHUINT, where x is one of the following:

- **HES0**: Combined interrupt output from Receiver frame of Host to External System {0-1} MHU 0
- **ESH0**: Combined interrupt output from Sender frame of External System {0-1} to Host MHU 0
- **HES1**: Combined interrupt output from Receiver frame of Host to External System {0-1} MHU 1
- **ESH1**: Combined interrupt output from Sender frame of External System {0-1} to Host MHU 1
- **SEES0**: Combined interrupt output from the Receiver frame of Secure Enclave to External System {0-1} MHU 0
- **ESSE0**: Combined interrupt output from the Sender frame of External System {0-1} to Secure Enclave MHU 0
- **SEES1**: Combined interrupt output from Receiver frame of Secure Enclave to External System {0-1} MHU 1
- **ESSE1**: Combined interrupt output from Sender frame of External System {0-1} to Secure Enclave MHU 1

The interface has the following properties:

- **EXTSYS{0-1}MHUCLK**
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}MHURESETn**

3.3.7 External System {0-1} Trace Expansion (EXTSYS{0-1}TRACEEXP) interface

The External System {0-1} Trace Expansion interface enables trace data to be routed into the SSE-700 debug infrastructure.

The interface has the following properties:

- ATB4
- 32-bit data
- **EXTSYS{0-1}ATCLK**
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}ATRESETn**

3.3.8 External System {0-1} Debug APB (EXTSYS{0-1}DBG) interface

The External System {0-1} Debug APB interface enables a debugger to access debug logic within the External System.

The interface has the following additional properties:

- APB4, including:
  - A DP Abort output
  - A **PWAKEUP** output
- 32-bit address and data
- **EXTSYS{0-1}DBGCLKM**
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}DBGPRESETMn**

3.3.9 External System {0-1} External Debug APB (EXTSYS{0-1}EXTDBG) interface

The External System {0-1} External Debug APB Slave interface enables the External System to access SSE-700 debug infrastructure.

The interface has the following properties:

- APB4, including a **PWAKEUP** input
- 32-bit address and data
- **EXTSYS{0-1}DBGCLKS**
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}DBGPRESETSn**

11.1.2 External Debug Bus memory map on page 11-169 defines the memory map for this interface.

3.3.10 External System {0-1} CTI Channel In (EXTSYS{0-1}CTICHIN) interface

The External System {0-1} CTI Channel In interface is an input CTI Channel interface. The interface enables the External System to connect its local CTIs to the CTI network provided by the SSE-700. This connection lets the External System drive triggers into SSE-700.

The interface has the following properties:

- CTI Channel interface
- **EXTSYS{0-1}CTICLK**
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}CTIRESETn**

3.3.11 External System {0-1} CTI Channel Out (EXTSYS{0-1}CTICHOUT) interface

The External System {0-1} CTI Channel Out interface is an output CTI Channel interface. The interface enables the External System to connect its local CTIs to the CTI network provided by the SSE-700. This connection lets the External System receive triggers from the SSE-700.

The interface has the following properties:

- CTI Channel interface
- **EXTSYS{0-1}CTICLK**
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}CTIRESETn**

3.3.12 External System {0-1} Shared Interrupt (EXTSYS{0-1}SHDINT) interface

The External System {0-1} Shared Interrupt interface is driven by ICI{2-3} of the Interrupt Router.

The interface has the following properties:

- Asynchronous
- AONTOP power domain
- Width is `NUM_EXP_SHD_INT` + 32

See *Table 9-1 Interrupt Router interface assignment* on page 9-150 for more information on Shared Interrupt number assignment.

3.3.13 External System {0-1} Memory Clock Q-Channel (EXTSYS{0-1}ACLKQ) interface

The External System {0-1} Memory Clock Q-Channel interface is a control Q-Channel interface that enables a clock controller in the External System to perform high-level clock gating of **EXTSYS{0-1}ACLK**.

The interface has the following properties:

- Q-Channel
- Asynchronous
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}ARESETn**

3.3.14 External System {0-1} MHU Clock Q-Channel (EXTSYS{0-1}MHUCLKQ) interface

The External System {0-1} MHU Clock Q-Channel interface is a control Q-Channel interface that enables a clock controller in the External System to perform high-level clock gating of **EXTSYS{0-1}MHUCLK**.

The interface has the following properties:

- Control Q-Channel
- Asynchronous
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}MHURESETn**

3.3.15 External System {0-1} Trace Clock Q-Channel (EXTSYS{0-1}ATCLKQ) interface

The External System {0-1} Trace Clock Q-Channel interface is a control Q-Channel interface that allows a clock controller in the External System to perform high-level clock gating of **EXTSYS{0-1}ATCLK**.

The interface has the following properties:

- Control Q-Channel
- Asynchronous
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}ATRESETn**

3.3.16 External System {0-1} Debug Master Clock Q-Channel (EXTSYS{0-1}DBGCLKMQ) interface

The External System {0-1} Debug Master Clock Q-Channel interface is a control Q-Channel interface that enables a clock controller in the External System to perform high-level clock gating of **EXTSYS{0-1}DBGCLKM**.

The interface has the following properties:

- Control Q-Channel
- Asynchronous
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}DBGPRESETMn**

3.3.17 External System {0-1} Debug Slave Clock Q-Channel (EXTSYS{0-1}DBGCLKSQ) interface

The External System {0-1} Debug Slave Clock Q-Channel interface is a control Q-Channel interface that enables a clock controller in the External System to perform high-level clock gating of **EXTSYS{0-1}DBGCLKS**.

The interface has the following properties:

- Control Q-Channel
- Asynchronous
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}DBGPRESETSn**

3.3.18 External System {0-1} CTI Clock Q-Channel (EXTSYS{0-1}CTICLKQ) interface

The External System {0-1} CTI Clock Q-Channel interface is a control Q-Channel interface that enables a clock controller in the External System to perform high-level clock gating of **EXTSYS{0-1}CTICLK**.

The interface has the following properties:

- Control Q-Channel
- Asynchronous
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}CTIRESETn**

3.3.19 External System {0-1} Memory Power Q-Channel (EXTSYS{0-1}MEMPWRQ) interface

The External System {0-1} Memory Power Q-Channel interface is a control Q-Channel interface. It enables the power controller in the External System to request an associated External System {0-1} Memory interface (**EXTSYS{0-1}MEM**) to leave or enter quiescent state for power control of the interface.

The interface has the following properties:

- Control Q-Channel
- Asynchronous
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}ARESETn**

3.3.20 External System {0-1} MHU Power Q-Channel (EXTSYS{0-1}MHUPWRQ) interface

The External System {0-1} MHU Power Q-Channel interface is a control Q-Channel interface. It enables the power controller in the External System to request an associated External System {0-1} MHU interface (**EXTSYS{0-1}MHU**) to leave or enter quiescent state for power control of the interface.

The interface has the following properties:

- Control Q-Channel
- Asynchronous
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}MHURESETn**

3.3.21 External System {0-1} MHU Power Request (EXTSYS{0-1}MHUPWRREQ) interface

The External System {0-1} MHU Power Request interface is a single signal, **EXTSYS{0-1}MHUPWRREQ**. This signal indicates when the Host System or Secure Enclave wants to send a message to the External System.

The interface has the following properties:

- Asynchronous
- AONTOP power domain
- **SYSTOPWARMRESETn/SECENCWARMRESETn**

  ———— **Note** ————

  This signal is a combination of signals from MHUs in the SYSTOP and SECENCTOP power domains.

  ————————————

3.3.22 External System {0-1} Trace Expansion Power Q-Channel (EXTSYS{0-1}TRACEEXPPWRQ) interface

The External System {0-1} Trace Expansion Power Q-Channel interface is a control Q-Channel interface. It enables the power controller in the External System to request an associated External System {0-1} Trace Expansion interface (EXTSYS{0-1}TRACEEXP) to enter or leave the quiescent state for power control of the interface.

The interface has the following properties:

- Control Q-Channel
- Asynchronous
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}ATRESETn**

3.3.23 External System {0-1} Debug APB Power Q-Channel (EXTSYS{0-1}DBGPWRQ) interface

The External System {0-1} Debug Power Q-Channel interface is a control Q-Channel interface. It enables the power controller in the External System to request an associated External System Debug interface (EXTSYS{0-1}DBG) to enter or leave quiescent state for power control of the interface.

The interface has the following properties:

- Control Q-Channel
- Asynchronous
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.3.24 External System {0-1} External Debug APB Power Q-Channel (EXTSYS{0-1}EXTDBGPWRQ) interface

The External System {0-1} External Debug APB Power Q-Channel interface is a control Q-Channel interface. It enables the power controller in the External System to request an associated External System {0-1} External Debug APB interface (EXTSYS{0-1}EXTDBG) to enter or leave quiescent state for power control of the interface.

The interface has the following properties:
- Control Q-Channel
- Asynchronous
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}DBGPRESETSn**

3.3.25 External System {0-1} CTI In Power Q-Channel (EXTSYS{0-1}CTIINPWRQ) interface

The External System {0-1} CTI Power Q-Channel interface is a control Q-Channel interface. It enables the power controller in the External System to request an associated CTI Channels In (EXTSYS{0-1}CTICHIN) to enter or leave quiescent state for power control of the CTI Channel interface.

The interface has the following properties:
- Control Q-Channel
- Asynchronous
- EXTSYS{0-1}TOP power domain
- **EXTSYS{0-1}CTIRESETn**

3.3.26 External System {0-1} CTI Out Power Q-Channel (EXTSYS{0-1}CTIOUTPWRQ) Interface

The External System {0-1} CTI Power Q-Channel interface is a control Q-Channel interface. It allows the power controller in the External System to request an associated CTI Channel Out (EXTSYS{0-1}CTICHOUT) to enter or leave quiescent state for power control of the CTI Channel interface.

The interface has the following properties:
- Control Q-Channel
- Asynchronous
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.3.27 External System {0-1} DBGTOP Q-Channel (EXTSYS{0-1}DBGTOPQ) interface

The External System {0-1} DBGTOP Q-Channel interface is a device Q-Channel interface that is controlled by a power controller for the DBGTOP domain.

The interface has the following properties:
- Device Q-Channel
- Asynchronous
- AONTOP power domain
- **AONTOPWARMRESETn**

3.3.28 External System {0-1} SYSTOP Q-Channel (EXTSYS{0-1}SYSTOPQ) interface

The External System {0-1} SYSTOP Q-Channel interface is a device Q-Channel interface that is controlled by a power controller for the SYSTOP domain.

The interface has the following properties:
- Device Q-Channel
- Asynchronous
- AONTOP power domain
- **AONTOPWARMRESETn**

3.3.29 External System {0-1} AONTOP Q-Channel (EXTSYS{0-1}AONTOPQ) interface

The External System {0-1} AONTOP Q-Channel interface is a device Q-Channel interface that is controlled by the Reset Controller.

The interface has the following properties:

- Device Q-Channel
- Asynchronous
- AONTOP power domain
- **AONTOPWARMRESETn**

3.3.30 External System {0-1} Power Request (EXTSYS{0-1}PWRREQ) interface

The External System {0-1} Power Request interface is formed from two CoreSight **CDBGPWRUPREQ**/**ACK** handshakes. One of the **CDBGPWRUPREQ**/**ACK** handshakes is from the EXTDBG ROM and the other is from the HOST AXIAP ROM.

The interface has the following properties:

- Formed of the following signals:
  - **EXTDBGROMCDBGPWRUPREQ**: Output
  - **EXTDBGROMCDBGPWRUPACK**: Input
  - **AXIAPROMCSYSPWRUPREQ**: Output
  - **AXIAPROMCSYSPWRUPACK**: Input
- Asynchronous
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.3.31 Reset Syndrome (EXTSYS{0-1}RSTSYN) interface

The Reset Syndrome interface is a 5-bit signal that indicates the cause of the last reset for the associated External System.

Note

Arm strongly recommends that this interface is exposed to software running on the External System, in a reset syndrome register.

The following table shows the bit assignment:

Table 3-7 EXTSYS{0-1}RSTSYN bit assignment

Bit	Name	Description
0	POR	Indicates the last reset of the External System was caused by one of the following: • **PORESETn** pin being asserted • DP **CDBGRSTREQ** being asserted • SoC Watchdog reset request • Secure Enclave Watchdog reset request • SOC_RST_CTRL.RST_REQ bit set to `0b1` • Secure Enclave software reset request
1	nSRST	Indicates that the last reset of the External System was caused by either: • **nSRST** pin being asserted • DP ROM **CSYSRSTREQ** being asserted
2	Reserved	Tied to `0b0`
3	HOST	Indicates the last reset of the External System was caused by a Host System reset request
4	EXT	Indicates the last reset of the External System was caused by a request to reset this External System

3.4 Host System Interfaces

This section describes the interfaces of the Host System, excluding those of the CPU_GIC socket.

This section contains the following subsections:

3.4.1 On-chip Volatile Memory (CVM) interface

There is a single AXI master expansion interface for on-chip volatile memory, for example SRAM.

The interface has the following properties:

- AXI5 with the following properties set to True:
  - Wakeup_Signals
  - Untranslated_transactions
- 32-bit address
- 64-bit data
- **ACLKOUT**
- SYSTOP power domain
- **SYSTOPWARMRESETn**

The CVM interface has a Firewall Component associated with it. For more information, see *8.3.5 Host System Firewall* on page 8-134.

This interface includes the AMBA AXI5 QoS Signals.

The CVM interface has an *Access Control Gate* (ACG), that moves SYSTOP to ON when there is an access to the On-chip Volatile Memory when the SYSTOP domain is not in the ON power-mode.

3.4.2 eXecute-in-place Non-volatile Memory (XNVM) interface

The *eXcute-in-Place* (XiP) Non-volatile Memory interface is an AXI Master interface. It provides access to an XiP Non-volatile memory, for example, Flash memory.

The interface has the following properties:

- AXI5 with the following properties set to True:
  - Wakeup_Signals
  - Untranslated_transactions
- 32-bit address

- 64-bit data
- **ACLKOUT**
- SYSTOP power domain
- **SYSTOPWARMRESETn**

The XNVM interface has a Firewall Component associated with it.

This interface includes the AMBA AXI5 QoS Signals.

3.4.3 Off-chip Volatile Memory (OCVM) interface

The Off-Chip Volatile Memory interface is an AXI Master interface. It provides access to off-chip volatile memory, for example, DRAM.

The interface has the following properties:

- AXI5 with the following properties set to true:
  - Wakeup_Signals
  - Untranslated_transactions
- 32-bit address
- 64-bit data
- **ACLKOUT**
- SYSTOP power domain
- **SYSTOPWARMRESETn**

The OCVM interface has a Firewall Component associated with it.

This interface includes the AMBA AXI5 QoS Signals.

3.4.4 Host Expansion Slave {0-1} (HOSTEXPSLV{0-1}) interfaces

There are two slave expansion interfaces of the Host System.

Each Interface has the following properties:

- AXI5 with the following properties set to True:
  - Wakeup_Signals
  - Untranslated_transactions
- 32-bit address
- 64-bit data
- **ACLKOUT**
- SYSTOP power domain
- **SYSTOPWARMRESETn**

Each HOSTEXPSLV{x} interface has a Firewall Component associated with it.

This interface includes the AMBA AXI5 QoS Signals.

3.4.5 Host Expansion Master {0-1} (HOSTEXPMST{0-1}) interfaces

There are two Host Expansion Master expansion interfaces for the Host System.

Each interface has the following properties:

- AXI5 with the following properties set to True:
  - Wakeup_Signals
  - Untranslated_transactions
- 32-bit address
- 64-bit data
- **ACLKOUT**
- SYSTOP power domain
- **SYSTOPWARMRESETn**

Each HOSTEXPMST{x} interface has a Firewall Component associated with it.

This interface includes the AMBA AXI5 QoS Signals.

3.4.6 Host AON Expansion Master (HOSTAONEXPMST) interface

The HOSTAONEXPMST interface allows integrators to add their own peripherals to the AONTOP domain.

This interface has the following properties:
- APB4, including a **PWAKEUP** output
- 32-bit address and data
- **HOSTAONEXPCLK**
- AONTOP power domain
- **AONTOPWARMRESETn**

3.4.7 Host Debug APB Expansion (HOSTDBGEXP) interface

The HOSTDBGAPBEXP interface allows integrators to add their own debug components to the Host System.

This interface has the following properties:
- APB4, including a **PWAKEUP** output
- 32-bit address and data
- **DBGCLKOUT**
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.4.8 Host Debug Trace Expansion (HOSTDBGTRACEEXP) interface

The HOSTDBGTRACEEXP interface allows integrators to add their own trace source components to the Host System.

This interface has the following properties:
- ATB4
- 32-bit data
- **DBGCLKOUT**
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.4.9 Host CTI Channel In Expansion (HOSTCTICHINEXP) interface

The Host CTI Channel In Expansion interface allows for an external CTI to connect and send trigger events to the internal CTI of the Host System.

This interface has the following properties:
- CTI Channel interface
- **DBGCLKOUT**
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.4.10 Host CTI Channel Out Expansion (HOSTCTICHOUTEXP) interface

The Host CTI Channel Out Expansion interface allows for an external CTI to connect and receive triggers from the internal CTI of the Host System.

This interface has the following properties:
- CTI Channel interface
- **DBGCLKOUT**
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.4.11 Host Debug Timestamp (HOSTTSVALUEB) interface

The CoreSight timestamp input interface is used by the CoreSight STM-500.

This interface has the following properties:
- 64-bit binary encoded timestamp
- **REFCLK**
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.4.12 Host Debug Authentication (HOSTDBGAUTH) interface

The Host Debug Authentication interface provides the CoreSight Authentication signals that must control any debug logic added to the Host System *Debug Authentication Zone* (DAZ).

The interface has the following properties:
- 4-bit signal made up of:
  - **DBGEN**
  - **NIDEN**
  - **SPIDEN**
  - **SPNIDEN**
- Asynchronous
- DBGTOP power domain
- **SEPORESETn**

3.4.13 Host STM DMA Peripheral Request (HOSTSTMDPR) interface

The Host STM DMA Peripheral Request interface enables communication between the STM and a DMA engine. It supports the DMA Peripheral Request interface, defined by CoreSight STM-500.

The interface has the following properties:
- CoreSight STM-500 DMA peripheral request interface. For more information on the signals and protocol, see the *Arm® CoreSight™ STM-500 System Trace Macrocell Technical Reference Manual*.
- **DBGCLK**
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.4.14 Host System REFCLK Generic Timestamp Gray (HOSTCNTVALUEG) interface

The Host System REFCLK Generic Timestamp Gray interface provides a Gray-encoded REFCLK application timestamp output. It can also be used by other components in the SoC added by the SoC integrator.

This interface has the following properties:
- 64-bit Gray encoded timestamp
- **HOSTCNTCLKOUT**
- AONTOP power domain
- **AONTOPWARMRESETn**

3.4.15 Host System REFCLK Generic Timestamp Binary (HOSTCNTVALUEB) interface

The Host System REFCLK Generic Timestamp binary interface provides an application timestamp value. This value is used for by the REFCLK timers {0-3} integrated in the SSE-700.

This interface has the following properties:
- 64-bit binary encoded timestamp
- **HOSTCNTCLKOUT**
- AONTOP power domain
- **AONTOPWARMRESETn**

3.4.16 Host System S32K Timestamp Gray (HOSTS32KCNTVALUEG) interface

The Host System 32K Generic Timestamp Gray interface provides a Gray-encoded 32K timestamp output. Connect it to the HOSTS32KCNTVALUEB interface to provide a timestamp for the S32K timers {0-1}. It can also be used by other components in the SoC, added by the integrator.

This interface has the following properties:
- 64-bit Gray encoded timestamp
- **HOSTS32CNTCLKOUT**
- AONTOP power domain
- **AONTOPWARMRESETn**

3.4.17 Host System S32K Timestamp Binary (HOSTS32KCNTVALUEB) interface

The Host System 32K Generic Timestamp binary interface provides the 32K timestamp, which is used by S32K timers {0-1}.

This interface has the following properties:
- 64-bit binary encoded timestamp
- **HOSTS32CNTCLKOUT**
- AONTOP power domain
- **AONTOPWARMRESETn**

3.4.18 Host System Debug Power Request (HOSTDBGPWRREQ) interface

The Host System Debug Power Request interface allows for additional power domains to be added to the Host System, to be controlled by a debug agent.

The interface has the following properties:
- 16 CoreSight Debug Power Request (Request/Acknowledge) handshakes from the Host AXIAP ROM CSYSPWRUP
- Asynchronous
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.4.19 Host System UART {0,1} (HOSTUART{0,1}) interface

The following table shows the Host System UART 0 and 1 interfaces, which provide a standard UART interface, with flow control.

Table 3-8 Host System UART interface signals

Signal name	Input/Output	Description
HOSTUART{0,1}TX	Output	UART Transmit data
HOSTUART{0,1}RX	Input	UART Receive data
HOSTUART{0,1}RTSn	Output	UART Request to Send
HOSTUART{0,1}CTSn	Input	UART Clear to Send
HOSTUART{0,1}RIn	Input	UART Ring Indicator
HOSTUART{0,1}DCDn	Input	UART Data Carrier Detect
HOSTUART{0,1}DSRn	Input	UART Data Set Ready
HOSTUART{0,1}DTRn	Output	UART Data Terminal Ready
HOSTUART{0,1}OUT1n	Output	UART Out1
HOSTUART{0,1}OUT2n	Output	UART Out2

The interface has the following properties:
- Asynchronous
- AONTOP power domain
- **AONTOPWARMRESETn**

3.5 Secure Enclave interfaces

The Secure Enclave interfaces are used to integrate a Crypto Accelerator into the SSE-700.

In the SSE-700, the Crypto Accelerator can be divided between the SECENCTOP and AONTOP power domains. The two pieces are called:

- *Crypto Accelerator* (CA)
- *Crypto Accelerator Always-On* (CA AON)

For detailed Secure Enclave Interface descriptions, see the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

It is IMPLEMENTATION DEFINED whether the Crypto Accelerator is implemented as whole or separate.

3.6 SSE-700 interfaces

This section describes the interfaces of the SSE-700.

This section contains the following subsections:

3.6.1 Clock inputs

SSE-700 has the following clock inputs:

- **S32KCLK**
- **REFCLK**
- **SECENCREFCLK**
- **SYSPLL**
- **CPUPLL**
- **SWCLKTCK**
- **TRACECLKIN**
- **UARTCLK**

3.6.2 Clock outputs

SSE-700 has the following clock outputs:

- **ACLKOUT**
- **DBGCLKOUT**
- **HOSTAONEXPCLK**
- **TRACECLKOUT**
- **HOSTCNTCLKOUT**
- **HOSTS32KCNTCLKCOUT**

3.6.3 Reset inputs

SSE-700 has the following reset inputs:

- **PORESETn**: Power-on reset of the whole SoC
- **nTRST**: Reset of the JTAG/SW part of the CoreSight DP
- **nSRST**: Debug reset of the whole SoC

SSE-700 treats these signals as asynchronous assert and deassert. An internal reset synchronizer is implemented for these signals to guarantee that the logic is released from reset synchronously.

3.6.4 Reset outputs

SSE-700 has the following reset outputs:

- **AONTOPPORESETn**
- **AONTOPWARMRESETn**
- **SYSTOPWARMRESETn**
- **DBGTOPWARMRESETn**

3.6.5 PLL Lock (PLLLOCK) interface

The PLL Lock interface is used to indicate when a PLL is locked and safe to use by software.

The PLLLOCK interface is formed of two signals:

- **CPUPLLLOCK**
- **SYSPLLLOCK**

Both signals are treated as asynchronous.

3.6.6 Expansion Shared Interrupt (EXPSHDINT) interface

The Expansion Shared Interrupt interface drives the SII interface of the Interrupt Router.

The interface has the following properties:

- `NUM_EXP_SHD_INT` bits wide
- Asynchronous
- AONTOP power domain
- **AONTOPWARMRESETn**

3.6.7 JTAG/Serial Wire Debug (SWJ) interface

SSE-700 has a JTAG/Serial Wire Debug interface which provides access to the CoreSight DP.

The interface has the following properties:

- Supports both JTAG and Serial Wire Debug protocols
- **SWCLKTCK**
- AONTOP power domain
- **nTRST/PORESETn**

  ———— **Note** ————

  When either reset is asserted, the SWJ interface is reset.

  ————————————

3.6.8 Trace Port Interface Unit (TPIU) interface

SSE-700 has a TPIU interface which enables trace data to be captured off-chip.

The interface is formed of the following signals:

- **TRACEDATA[31:0]**
- **TRACECTL**
- **TRACEMAXDATASIZE[4:0]**
- **TPCTLVALID**

For more information on the interface, see the *Arm CoreSight™ System-on-Chip SoC-600 Technical Reference Manual*.

The width of the TPIU interface can be configured to any number of bits, starting with **TRACEDATA[0]**, using the **TRACEMAXDATASIZE[4:0]** signal.

The interface has the following properties:

- **TRACECLKOUT**
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.6.9 SoC Configuration (SOCCFG) Interface

SOCCFG is a static configuration interface. The signals are driven from the AONTOP domain.

The interface is formed of the following signals:

- **CVMSIZE[7:0]**
- **XNVMSIZE[7:0]**
- **OCVMSIZE[7:0]**

This interface has the following properties:

- **S32KCLK**
- AONTOP power domain
- **AONTOPWARMRESETn**

3.6.10 SoC Identification (SOCID) interface

The SOCID interface is used to set the SoC Identification information in the System ID registers.

The signals are considered pseudo static and are driven from the AONTOP domain.

The interface is formed of the following signals:

- **SOCPRTID[11:0]**: SoC Product ID
- **SOCVAR[3:0]**: SoC Variant
- **SOCREV[3:0]**: SoC Revision
- **SOCIMPLID[10:0]**: SoC Implementer JEP106 Code ID:
  - [6:0]: JEP106 identity code
  - [10:7]: JEP106 continuation code
- **DPROMPRTID[11:0]**: DP ROM table Part ID
- **DPROMVAR[3:0]**: DP ROM table Variant
- **DPROMREV[3:0]**: DP ROM table Revision
- **DPROMIMPLID[10:0]**: DP ROM table implementor JEP106 Code ID:
  - [6:0]: JEP106 identity code
  - [10:7]: JEP106 continuation code
- **EXTDBGROMPRTID[11:0]**: EXTDBG ROM table Part ID
- **EXTDBGROMVAR[3:0]**: EXTDBG ROM table Variant
- **EXTDBGROMREV[3:0]**: EXTDBG ROM table Revision
- **EXTDBGROMIMPLID[10:0]**: EXTDBG ROM table implementor JEP106 Code ID:
  - [6:0]: JEP106 identity code
  - [10:7]: JEP106 continuation code
- **HOSTROMPRTID[11:0]**: Host ROM table Part ID
- **HOSTROMVAR[3:0]**: Host ROM table Variant
- **HOSTROMREV[3:0]**: HOST ROM table Revision
- **HOSTROMIMPLID[10:0]**: HOST ROM table implementor JEP106 Code ID:
  - [6:0]: JEP106 identity code
  - [10:7]: JEP106 continuation code
- **HOSTAXIAPROMPRTID[11:0]**: Host AXIAP ROM table Part ID
- **HOSTAXIAPROMVAR[3:0]**: Host AXIAP ROM table Variant
- **HOSTAXIAPROMREV[3:0]**: Host AXIAP ROM table Revision
- **HOSTAXIAPROMIMPLID[10:0]**: Host AXIAP ROM table implementor JEP106 Code ID:
  - [6:0]: JEP106 identity code
  - [10:7]: JEP106 continuation code

3.6.11 SoC Security Control (SOCSC) interface

The SOCSC interface contains miscellaneous control signals for controlling SoC features.

The SOCSC interface is formed of the following output signals:

- **DFTENABLE[1:0]**:

  - Bit[0] is driven by bit[34] of the SCB interface. An active-HIGH signal, which indicates when Design for Test functionality can be enabled on the SoC, except for logic in the Secure Enclave. For example, scan or MBIST.
  - Bit[1] is driven by bit[63] of the SCB interface. An active-HIGH signal, which indicates when Design for Test functionality can be enabled on the logic inside the Secure Enclave. For example, scan or MBIST.
  - Both of these signals must not prevent access to the debug DP, in the SSE-700.
- **SCBEXP[63:0]**:
  - Driven by bits[127:64] of the SCB interface.
  - Behavior of these signals is defined by the integrator.

For more information on the SCB, see the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

This interface has the following properties:
- Asynchronous
- AONTOP power domain
- **SEPORESETn**

3.6.12 SoC Lifecycle Control (SOCLCC) interface

The SoC Lifecycle Control interface is an input pin, which controls the lifecycle state of the SoC.

This interface has the following properties:
- Asynchronous
- AONTOP power domain

3.7 Clock Control interfaces

The SSE-700 provides several Q-Channel interfaces to support high-level clock gating of either clock inputs or outputs.

This section contains the following subsections:

3.7.1 REFCLK Q-Channel (REFCLKQ) interface

The REFCLK Q-Channel is a Q-Channel control interface that allows high-level clock gating of the **REFCLK**. All signals have the format **REFCLK{x}**, where x is the standard Q-Channel protocol name.

This interface has the following properties:

- Control Q-Channel
- Asynchronous
- AONTOP power domain
- **PORESETn**

3.7.2 ACLK Q-Channel (ACLKQ) interface

The ACLK Q-Channel interface is formed of one or more device Q-Channels.

This interface has the following properties:

- (`NUM_ACLK_QCH` + 7) device Q-Channels
- Asynchronous
- SYSTOP power domain
- **SYSTOPWARMRESETn**

3.7.3 DBGCLK Q-Channel (DBGCLKQ) interface

The DBGCLK Q-Channel interface is formed of one or more device Q-Channels.

This interface has the following properties:

- NUM_DBGCLK_QCH device Q-Channels
- Asynchronous
- DBGTOP power domain
- **DBGTOPWARMRESETn**

3.8 Power Control interfaces

This section gives an overview of the Power Control interfaces.

This section contains the following subsections:

3.8.1 SYSTOP Q-Channel (SYSTOPQ) interface

The SYSTOPQ interface enables expansion of the SYSTOP power domain.

This interface has the following properties:

- Formed of 3 device Q-Channels:
  - Bit 0 is Egress Q-Channel
  - Bit 1 is Internal Q-Channel
  - Bit 2 is Ingress Q-Channel
- Asynchronous
- AONTOP power domain
- **AONTOPWARMRESETn**

3.8.2 DBGTOP Q-Channel (DBGTOPQ) interface

The DBGTOPQ interface enables expansion of the DBGTOP power domain and

This interface has the following properties:

- Formed of 3 device Q-Channels:
  - Bit 0 is Egress Q-Channel
  - Bit 1 is Internal Q-Channel
  - Bit 2 is Ingress Q-Channel
- Asynchronous
- AONTOP power domain
- **AONTOPWARMRESETn**

Chapter 4
Clocks

This chapter describes the primary and internal clocks within the SSE-700.

It contains the following sections:

4.1 Clock inputs

The SSE-700 has the following clock inputs:

- **REFCLK**: Required
- **S32KCLK**: Required
- **SECENCREFCLK**: Required
- **CPUPLL**: Optional
- **SYSPLL**: Required
- **SWCLKTCK**: Required
- **TRACECLKIN**: Required
- **UARTCLK**: Required
- External System Clock Inputs: Required:
  - **EXTSYS{0-1}DBGCLKS**
  - **EXTSYS{0-1}DBGCLKM**
  - **EXTSYS{0-1}ATCLK**
  - **EXTSYS{0-1}CTICLK**
  - **EXTSYS{0-1}ACLK**
  - **EXTSYS{0-1}MHUCLK**

——— **Note** ———

The integrator can drive multiple clock inputs from the same clock source, depending on SoC requirements.

———————————

This section contains the following subsections:

4.1.1 REFCLK

REFCLK is the reference clock for the system, except for the Secure Enclave. It is used in many places and is the default option for all generated clocks.

The REFCLK Q-Channel interface indicates when **REFCLK** is required by SSE-700. For more information, see *3.7.1 REFCLK Q-Channel (REFCLKQ) interface* on page 3-56. **REFCLK** can be gated when the Q-Channel enters the Q_STOPPED state. When QACTIVE is asserted **REFCLK** must be ungated and the Q-Channel returned to the Q_RUN state.

4.1.2 S32KCLK

S32KCLK is used by the S32K counter, timers, and SoC watchdog in the system. It is also used by the wakeup logic when in the BSYS.SLEEP1 power state.

4.1.3 SECENCREFCLK

SECENCREFCLK is the reference clock for the Secure Enclave. For details, see *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

4.1.4 CPUPLL

CPUPLL is used to generate the clocks used by the Host CPU. The **CPUPLL** input must be from a PLL, added by the integrator.

There is also a **CPUPLLLOCK** signal.

The software is responsible for switching to the **CPUPLL** when the PLL provides a clock and is locked to the frequency.

——— **Note** ———

If the CPU PLL is present but does not provide a lock signal, then it is IMPLEMENTATION DEFINED how the software determines when the PLL is stable enough to be used.

———————————

4.1.5 SYSPLL

SYSPLL is the clock output of the system PLL. It generates all clocks in the system. Alongside the **SYSPLL** input, there is the **SYSPLLLOCK** signal.

The software is responsible for switching to the **SYSPLL** only when the PLL provides a clock and is locked to the frequency.

——— **Note** ———

If the System PLL does not provide a lock signal, then it is IMPLEMENTATION DEFINED how the software determines when the PLL is stable enough to be used.

———————————

4.1.6 TRACECLKIN

TRACECLKIN is the clock that drives the TPIU.

TRACECLKIN is required when trace is required to be exported by the TPIU. The integrator needs to ensure that **TRACECLKIN** is provided in the SoC.

4.1.7 SWCLKTCK

SWCLKTCK is the clock that drives the JTAG and *Serial Wire Debug* (SWD) interface.

4.1.8 UARTCLK

The **UARTCLK** is the clock that generates the **HOSTUARTCLK** that is used by the Host System UARTs.

The Host System UART has a programmable baud rate. The maximum baud rate is max(**REFCLK**, **S32KCLK**, **UARTCLK**)/16. For more information, see the *Arm® PrimeCell® UART (PL011) Technical Reference Manual*.

The **UARTCLK** is always available when SSE-700 is in the BSYS.RUN and BSYS.SLEEP0 power states. SSE-700 supports both **UARTCLK** and **S32KCLK** as source for UARTs when SSE-700 is in SLEEP1 power state to enable wakeup from either of the Host UARTs. If the **UARTCLK** is not provided in the BSYS.SLEEP1 state and wakeup from either Host UART is required, software must select the **S32KCLK** as the source for the **HOSTUARTCLK**.

4.1.9 External System

The following clock inputs are provided for logic inside the External System Harness:

- **EXTSYS{0-1}DBGCLKS**: Clock used for the EXTSYS{0-1}DBG interface
- **EXTSYS{0-1}DBGCLKM**: Clock used for the EXTSYS{0-1}EXTDBG interface
- **EXTSYS{0-1}ATCLK**: Clock used for the EXTSYS{0-1}TRACEEXP interface
- **EXTSYS{0-1}CTICLK**: Clock used for the EXTSYS{0-1}CTICHIN/EXTSYS{0-1}CTICHOUT interfaces

- **EXTSYS{0-1}ACLK**: Clock used for the EXTSYS{0-1}MEM interface
- **EXTSYS{0-1}MHUCLK**: Clock used for the EXTSYS{0-1}MHU interface

Each of these clocks support high-level clock gating and a corresponding clock Q-Channel is provided.

All the EXTSYS{0-1}{x} clocks must be provided whenever the clock Q-Channel **QACTIVE** is HIGH.

4.2 Internal clocks

There are several internal clocks within the SSE-700:

- **ACLK**
- **CTRLCLK**
- **GICCLK**
- **DBGCLK**
- **SECENCCLK**
- **SECENCDIVCLK**
- **HOSTCPUCLK**

The following figure shows the clock domain mapped over the SSE-700 topology diagram.

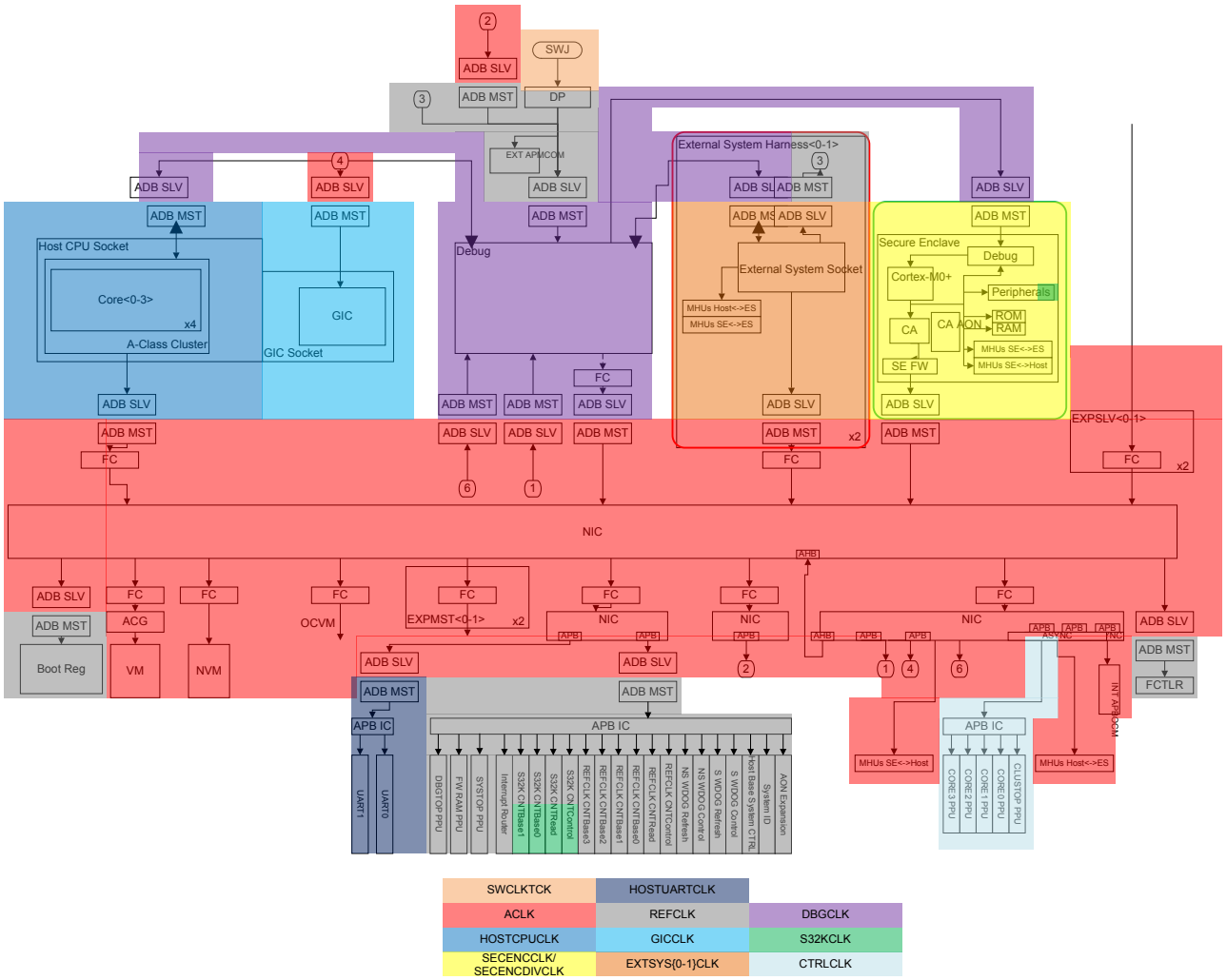

Figure 4-1 Clock domains of SSE-700

This section contains the following subsections:

- *4.2.8 S32KCLK* on page 4-65.
- *4.2.9 HOSTUARTCLK* on page 4-66.

4.2.1 ACLK

The following table describes the **ACLK** clock.

Table 4-1 ACLK summary table

Clock name	**ACLK**		
Sources	Name	Default	Divider support
	REFCLK	Yes	No
	SYSPLL	No	Yes
High-level clock gating support	Yes		
Components	• NIC(Main) • NIC(AONPERIPH) • NIC(SYSPERIPH) • NIC(DBGPERIPH) • The following are Firewall components of the Host System Firewall: — SYSPERIPH — DBGPERIPH — AONPERIPH — XNVM — CVM — HOSTCPU — EXTSYS{0-1} — EXPSLV{0-1} — EXPMST{0-1} — OCVM • Host System side of MHUs with External Systems and Secure Enclave • INT APBCOM		

4.2.2 GICCLK

The following table describes the **GICCLK** clock.

Table 4-2 GICCLK summary table

Clock name	**GICCLK**		
Sources	Name	Default	Divider Support
	REFCLK	Yes	No
	SYSPLL	No	Yes
High-level clock gating support	No		
Components	GIC		

4.2.3 DBGCLK

The following table describes the **DBGCLK** clock.

Table 4-3 DBGCLK summary table

Clock name	**DBGCLK**		
Sources	Name	Default	Divider support
	REFCLK	Yes	No
	SYSPLL	No	Yes
High-level clock gating support	Yes		
Components	• Host ETR • Host CATU • Host STM • Host CTI • Host Funnel • Host Replicator • Host AXI AP • Host APB AP • Host AXIAP ROM • Host ROM • Host CTM • SoC CTM • SoC TPIU Replicator • SoC TPIU • SoC ETR • SoC CATU • Counter CTI • Debug Firewall Component of the Host System Firewall • Channel Gates for the following DAZs: — HOSTAUTH — TPIUAUTH — COUNTERAUTH		

4.2.4 SECENCCLK and SECENCDIVCLK

For a description about the **SECENCCLK** and **SECENCDIVCLK** clocks, see the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

4.2.5 HOSTCPUCLK

The following table describes the **HOSTCPUCLK** clock.

Table 4-4 HOSTCPUCLK summary table

Clock name	**HOSTCPUCLK**		
Sources	Name	Default	Divider support
	REFCLK	Yes	No
	SYSPLL	No	Yes
	CPUPLL	No	Yes
High-level clock gating support	No		
Components	Host CPU		

HOSTCPUCLK is provided as an output clock, **HOSTCPUCLKOUT**, which is only used within the CLUSTOP and CORE{x} power domains.

4.2.6 CTRLCLK

The following table describes the **CTRLCLK** clock.

Table 4-5 CTRLCLK summary table

Clock name	CTRLCLK		
Sources	Name	Default	Divider support
	REFCLK	Yes	No
	SYSPLL	No	Yes
High-level clock gating support	Yes		
Components	• CORE{0-3} PPU • CLUSTOP PPU		

4.2.7 REFCLK

The following table describes the **REFCLK** clock.

Table 4-6 REFCLK summary table

Clock name	REFCLK
Source	**REFCLK**
High-level clock gating support	Yes
Components	• Interrupt Router • REFCLK CNTBase{0-3} • REFCLK CNTRead • REFCLK CNTControl • S32K CNTBase{0-1} • S32K CNTRead • S32K CNTControl • Non-secure Watchdog Refresh and Control • Secure Watchdog Refresh and Control • Host Base System Control • System ID • HOSTAONEXPMST • Firewall Controller of the Host System Firewall • DBGTOP PPU • Firewall PPU • SYSTOP PPU • Boot Register

The **REFCLK** generates the **HOSTAONEXPCLK** and **HOSTCNTCLKOUT** output clocks.

4.2.8 S32KCLK

The following table describes the **S32KCLK** clock.

Table 4-7 S32KCLK summary table

Clock name	S32KCLK
Source	**S32KCLK**
High-level clock gating support	No
Components	• S32K CNTBase{0-1} • S32K CNTRead • S32K CNTControl • SoC Watchdog

The **S32KCLK** generates the **HOSTS32KCNTCLKOUT** output clock.

4.2.9 HOSTUARTCLK

The following table describes the **HOSTUARTCLK** clock.

Table 4-8 HOSTUARTCLK summary table

Clock name	HOSTUARTCLK		
Sources	Name	Default	Divider support
	REFCLK	Yes	No
	UARTCLK	No	Yes
	S32KCLK	No	No
High-level clock gating support	No		
Components	Host UART {0,1}		

4.3 Output clocks

SSE-700 provides the following clock outputs:

- **ACLKOUT**:
  - Driven by **ACLK**
  - Used only in SYSTOP
- **GICCLKOUT**:
  - Driven by **GICCLK**
  - Used only in CLUSTOP
- **DBGCLKOUT**:
  - Driven by **DBGCLK**
  - Used only in DBGTOP
- **HOSTCPUCLKOUT**:
  - Driven by **HOSTCPUCLK**
  - Used only in CLUSTOP
- **CRYPTOCLKOUT**:
  - Driven by **SECENCCLK**
  - Used only to integrate the Crypto Accelerator into the SECNECTOP part of the Secure Enclave
- **CRYPTOAONCLKOUT**:
  - Driven by **SECENCCLK**
  - Used only to integrate the Crypto Accelerator Always-on into the AONTOP part of the Secure Enclave
- **HOSTAONEXPCLK**:
  - Driven by **REFCLK**
  - Used only for logic connected to the **HOSTAONEXPMST**
- **TRACECLKOUT**:
  - Driven by the **TPIU**
  - Half the frequency of **TRACECLKIN**
  - Used only for the TPIU interface
- **HOSTCNTCLKOUT**:
  - Driven by **REFCLK**
  - Used only for distribution of, and any additional timers added to, the **REFCLK** timestamp, provided by **HOSTTSVALUEG**
- **HOSTS32KCNTCLKOUT**:
  - Driven by **S32KCLK**
  - Used only for distribution of, and any additional timers added to, the S32K timestamp, provided by **HOSTS32KCNTVALUEG**

Chapter 5
Power

This chapter describes the voltage and power domains within the SSE-700.

It contains the following sections:

5.1 Power overview

The SSE-700 supports multiple voltage and power domains.

Each voltage domain has at least one power domain, and each power domain supports several power modes.

5.2 PPU

Each power domain is controlled by a PCK-600 PPU that conforms to *Arm® Power Policy Unit Architecture Specification, version 1.1*. The PPU is responsible for controlling the power domain.

All PPUs support the following power modes: WARM_RST, ON, and OFF. Some PPUs support extra power modes.

PPUs have two modes of operation:

Static mode transitions

To calculate whether to remain in the current power mode or transition to the policy mode, the PPU uses:

- The policy in the PWR_POLICY register
- Hardware indications from components inside the power domain that use the **PACTIVE** or **QACTIVE** signals

In this mode of operation, the PPU can only transition to the programmed policy mode.

Dynamic mode transitions

The PPU uses the policy and hardware indications to calculate the lowest power mode the domain must be in. In this mode of operation, the PPU can transition to any of the required power modes autonomously without direct software intervention.

Note

Arm strongly recommends that dynamic modes are enabled for all PPUs integrated within the SSE-700 subsystem by default. Software is not expected to require direct access to the PPU after performing an initial configuration at boot. Therefore, software is expected to use the Host CPU Cluster Power State register and Base System Power Request register to control the SSE-700 power modes and state power modes.

The PPU interrupts, excluding the SECENCTOP PPU, are combined to form a single interrupt called the Host PPU Combined interrupt. This interrupt is used by the Host CPU or Secure Enclave depending on the settings of the Interrupt Router. Software uses the HOST_PPU_INT_ST register to identify which PPU has generated the interrupt. The interrupt from the SECENCTOP PPU is connected only to the Secure Enclave Cortex-M0+ NVIC.

For more information on:

- The PCK-600 PPU, see the *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual*.
- The PPU v1.1 architecture, see the *Arm® Power Policy Unit Architecture Specification, version 1.1*.
- The expected software usage of the PPU in the SSE-700, see *12.1 Power control* on page 12-239.

Note

Arm strongly recommends that software:

- Does not set the PPU to static mode transitions. This can lead to loss of functionality or deadlock of the SoC.
- Never programs the policy of the PPU to WARM_RST. This can lead to deadlock of the SoC.

This section contains the following subsections:

5.2.1 PPU configurations

The PPUs within the SSE-700 are configured with the following:

- P-Channels for the device interfaces
- Default power mode of dynamic OFF
- All power modes are enabled for both static and dynamic transitions, unless stated otherwise in the following sections.
- No support for operating modes
- Support power mode entry delay, except Core{0-3} and CLUSTOP PPUs

5.2.2 PPU debug

The PPU includes the registers PPU_PWCR and PPU_PTCR for debugging power issues during SoC development.

Bit[35] of the *Security Control Bits* (SCB) interface controls whether these registers are read/write or are treated as WI and generate an error. When the SCB is HIGH, writes to the registers are allowed. For more information on the SCBs, see *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

——— **Note** ———

Using PPU_PWCR or PPU_PTCR can lead to UNPREDICTABLE results depending on the power transitions being attempted. Arm strongly recommends that you use these registers for debugging purposes only.

———————————

5.3 Voltage domains

SSE-700 has the following voltage supplies:

- VSYS:
  - Voltage supply for AONTOP, SYSTOP, SECENCTOP, and DBGTOP power domains
  - Voltage supply is always available, unless SoC is in BSYS.OFF power state.
- VCLUS:
  - Voltage supply for CLUSTOP and CORE{0-3} power domains
  - DVFS supported
  - Voltage supply is automatically enabled when domain requires it without any software intervention.
- VEXTSYS{0-1}:
  - Voltage supply for EXTSYS{0-1}TOP power domain
  - DVFS supported
  - Voltage supply is automatically enabled when domain requires it without any software intervention.

It is legal for an implementation to drive all voltages from a single voltage supply, VSYS.

5.4 Power domains

The following figure shows the power domains that are supported by the SSE-700. The power domains are mapped over a topology diagram of the SSE-700.

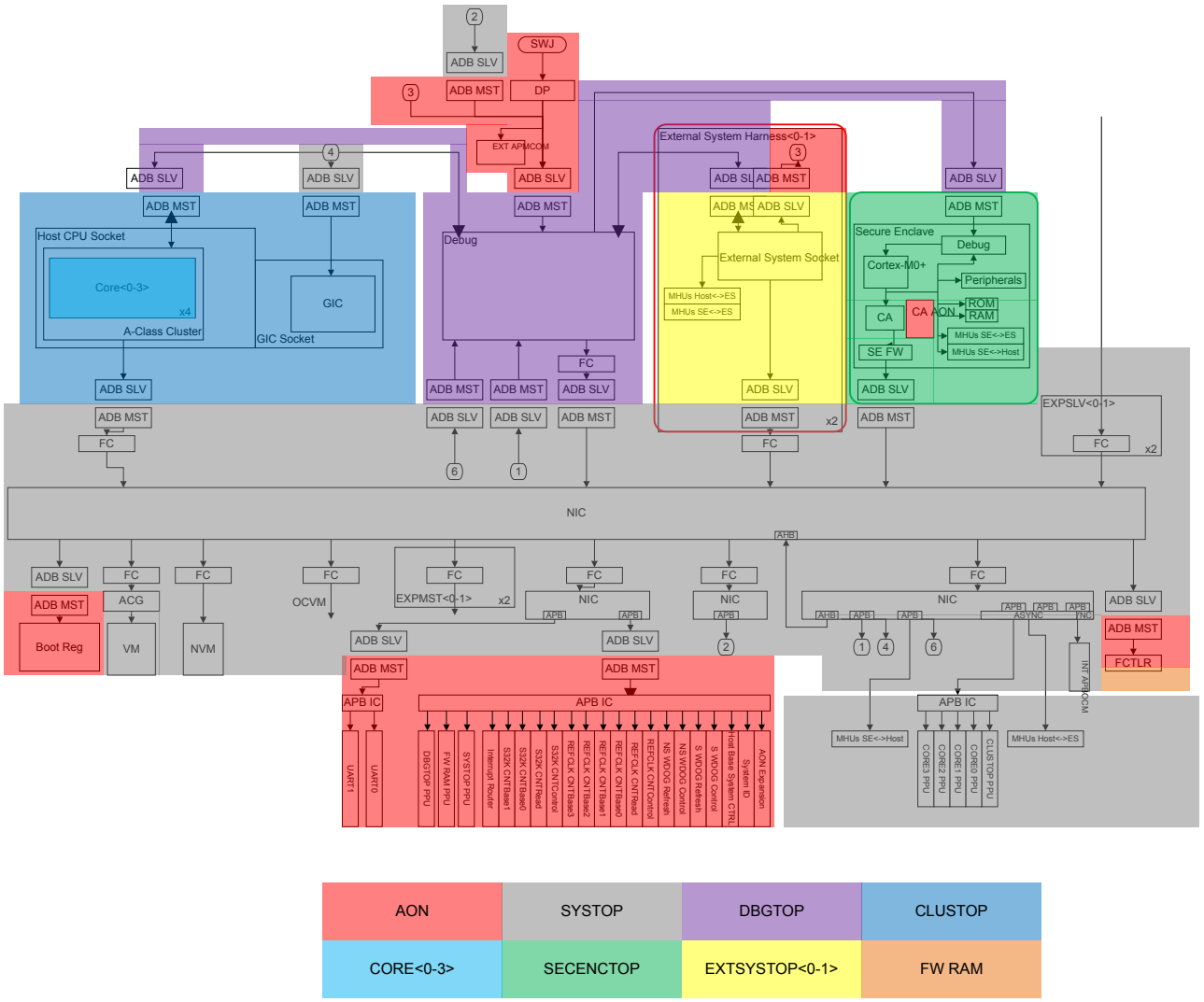

Figure 5-1 Power domains of SSE-700

The following subsections provide information about:

- The components that are contained within each power domain
- The power modes that are supported by the domain
- The conditions on interfaces into and out of the domain, when in a specific power mode

This section contains the following subsections:

5.4.1 AONTOP

AONTOP is the always-on power domain in the system.

AONTOP contains the following components:

- Generic counters for REFCLK and 32K time domains
- Generic timers for both REFCLK and 32K time domains
- Secure and Non-secure Watchdogs
- Interrupt Router
- Host Base System Control
- System ID
- Host System Firewall Controller
- CoreSight DP and ROM table
- EXT APBCOM
- GPIO Control
- Wakeup logic
- Firewall, SYSTOP, DBGTOP PPUs, and PCSMs.
- Firewall Controller
- Boot register
- PCSM of the CLUSTOP PPU

  ———— **Note** ————

  This does not include the CLUSTOP PPU itself.

  ————————————

- SoC Watchdog
- Secure Enclave UART
- SECENCTOP PPU and PCSM
- Crypto Accelerator Always-on
- Secure Enclave System and Base System Control

The AONTOP domain is not controlled by a PPU. Instead, the logic is always powered whenever VSYS is applied.

Firewall RAM

The Firewall PPU only controls the Host System Firewall shadow registers. These registers are implemented as RAM.

The Firewall PPU supports the following power modes:

- WARM_RST: It has no effect on the Host System Firewall and is considered the same as ON.
- ON: The Host System Firewall shadow registers are powered.
- FUNC_RET: The Host System Firewall shadow registers are in a retention state.
- OFF: The Host System Firewall shadow registers are not powered.

The following applies when Firewall PPU is in, or attempts to enter, the following modes:

- OFF power mode: The software must never request entry into the OFF power mode.
- FUNC_RET power mode: Any access to the shadow registers of the Host System Firewall is stalled.
- ON power mode: Any access to the shadow registers of the Host System Firewall is allowed.
- WARM_RST mode: The software must never request entry into the WARM_RST power mode.

5.4.2 SYSTOP

SYSTOP contains the following components:

- NIC-450 (Main/AONPERIPH/SYSPERIPH/DBGPERIPH)
- INT APBCOM
- Sender frames of the following MHUs, if they are present:
  — HSE{0-1}
  — HES{0-1}{0-1}

- Receiver frames of the following MHUs, if they are present:
  - SEH{0-1}
  - ES{0-1}H{0-1}
- The following Host System Firewall components:
  - SYSPERIPH
  - DBGPERIPH
  - AONPERIPH
  - XNVM
  - CVM
  - HOSTCPU
  - EXTSYS{0-1}
  - EXPSLV{0-1}
  - EXPMST{0-1}
  - OCVM
- CLUSTOP and CORE{0-3} power control logic: PPU and PCSM except for CLUSTOP PCSM which is in AONTOP.

Note

The SYSTOP domain is further split into logic and memory subdomains. The memory domain only contains the on-chip volatile memory. All other logic is in the logic subdomain.

The SYSTOP power domain is controlled by the SYSTOP PPU, which supports the following power modes:

- WARM_RST: SYSTOP is powered ON but all logic is held in reset.
- ON: Logic and the volatile memory are powered.
- FUNC_RET: Logic is powered, but the volatile memory is in retention.
- MEM_RET: Logic is not powered, but the volatile memory is in retention.
- OFF: Logic and volatile memory are not powered.

The following applies when the SYSTOP domain is in the following modes:

- OFF power mode:
  - Any memory accesses from following masters or interfaces, listed below, to the Host System address space are stalled and a wakeup is generated to enter the FUNC_RET power mode:
    - Host ETR and CATU
    - SoC ETR and CATU
    - AXI-AP
    - External System {0-1}
    - Secure Enclave
  - BSYS_PWR_ST.SYSTOP_PWR_ST field is set to `0b000` in both the Host and Secure Enclave Base System Control registers.
- MEM_RET power mode:
  - Any memory access from the following masters or interfaces, listed below, to the Host System address space is stalled and a wakeup is generated to enter the FUNC_RET power mode:
    - Host ETR and CATU
    - SoC ETR and CATU
    - AXI-AP

    - External System {0-1}
    - Secure Enclave
  - BSYS_PWR_ST.SYSTOP_PWR_ST field is set to `0b001` in both the Host and Secure Enclave Base System Control registers.
- FUNC_RET power mode:
  - Any memory access to the On-chip Volatile Memory is stalled at the ACG on the CVM interface, and a request is generated to enter the ON power mode.
  - Any memory access from the following masters or interfaces to the Host System address space is allowed:
    - Host CPU
    - Host GIC
    - Host ETR and CATU
    - SoC ETR and CATU
    - AXI-AP
    - External System {0-1}
    - Secure Enclave
  - BSYS_PWR_ST.SYSTOP_PWR_ST field is set to `0b010` in both the Host and Secure Enclave Base System Control registers.
- ON power mode:
  - Any memory access to the On-chip Volatile Memory is allowed through the ACG on the CVM interface.
  - Any memory access from the following masters or interfaces to the Host System address space is allowed:
    - Host CPU
    - Host GIC
    - Host ETR and CATU
    - SoC ETR and CATU
    - AXI-AP
    - External System {0-1}
    - Secure Enclave
  - BSYS_PWR_ST.SYSTOP_PWR_ST field is set to `0b100` in both the Host and Secure Enclave Base System Control registers.
- WARM_RST power mode:
  - Software must never request entry into the WARM_RST power mode because it cannot cause the SYSTOP power domain to exit the mode.
  - Any memory access to the On-chip Volatile Memory is stalled at the ACG.
  - BSYS_PWR_ST.SYSTOP_PWR_ST field is set to `0b000` in both the Host and Secure Enclave Base System Control registers.

The SYSTOP power domain can be extended by an implementation, using the Q-Channels of the SYSTOPQ interface. For more information on the SYSTOPQ interface, see *3.8.1 SYSTOP Q-Channel (SYSTOPQ) interface* on page 3-57.

5.4.3 DBGTOP

The DBGTOP power domain contains the following components:

- Host APB AP
- Host AXI AP
- Host AXIAP ROM
- Host ROM
- Host ETR
- Host CATU
- Host STM
- Host CTI
- Host Funnel
- Host Replicator

- Host CTM
- SoC TPIU Funnel
- SoC TPIU Replicator
- SoC TPIU
- SoC CTI
- SoC ETR
- SoC CATU
- SoC CTM
- Counter CTI
- EXTDBG ROM
- Debug Firewall Component, of the Host System Firewall
- Channel Gates for the following DAZs:
  - HOSTAUTH
  - TPIUAUTH
  - COUNTERAUTH

The DBGTOP power domain is controlled by the DBG PPU, which supports the following power modes:

- WARM_RST: Reset all logic in the domain.
- ON: Debug logic is powered.
- OFF: Debug logic is not powered.

The following applies when the DBGTOP domain is in the:

- OFF power mode:
  - Any access to the Host System Debug, from any debug agent, returns an error response.
  - Any access to the External Debug Bus, except for the DP ROM or EXT APBCOM, from any debug agent, returns an error response.
  - Any access to the CoreSight STM-500 Extended Stimulus Port is treated as RAZ/WI with no error response.
  - Any trace data from the following interfaces is ignored:
    - HOSTCPUTRACE
    - EXTSYS{0-1}TRACEEXP
    - HOSTDBGTRACEEXP.
  - Any CTI events from the following interfaces or components are ignored:
    - HOSTCPUCTICH{IN,OUT}
    - EXTSYS{0-1}CTICH{IN,OUT}
    - HOSTCTICH{IN,OUT}EXP
    - Secure Enclave CTI
  - BSYS_PWR_ST.DBGTOP_PWR_ST is `0b0` in both the Host and Secure Enclave Base System Control registers.
- ON power mode:
  - Any access to the Host System Debug is allowed from any debug agent.
  - Any access to the External Debug Bus is allowed.
  - Any trace data from the following interfaces is accepted:
    - HOSTCPUTRACE
    - EXTSYS{0-1}TRACEEXP
    - HOSTDBGTRACEEXP
  - Any CTI events are allowed on the following interfaces or components:
    - HOSTCPUCTICH{IN,OUT}
    - EXTSYS{0-1}CTICH{IN,OUT}
    - HOSTCTICH{IN,OUT}EXP
    - Secure Enclave CTI

  - BSYS_PWR_ST.DBGTOP_PWR_ST is `0b1` in both the Host and Secure Enclave Base System Control registers.
  - Accesses to the Host System memory map are allowed.
- WARM_RST power mode:
  - Any access to the Host System Debug, from any debug agent, returns an error response.
  - Any access to the External Debug Bus, except for the DP ROM or EXT APBCOM, from any debug agent, returns an error response.
  - Any access to the CoreSight STM-500 Extended Stimulus Port is treated as RAZ/WI with no error response.
  - Any trace data from the following interfaces is ignored:
    - HOSTCPUTRACE
    - EXTSYS{0-1}TRACEEXP
    - HOSTDBGTRACEEXP
  - Any CTI events are ignored from the following interfaces or components:
    - HOSTCPUCTICH{IN,OUT}
    - EXTSYS{0-1}CTICH{IN,OUT}
    - HOSTCTICH{IN,OUT}EXP
    - Secure Enclave CTI
  - BSYS_PWR_ST.DBGTOP_PWR_ST is `0b0` in both the Host and Secure Enclave Base System Control registers.

The debug agent is required to power on the DBGTOP domain, before attempting to configure or use the debug logic in the DBGTOP power domain, using one of the following methods:

- Setting the DP ROM **CDBGPWRUPREQ0** HIGH and waiting for **CDBGPWRUPACK0** to go HIGH
- Setting the BSYS_PWR_REQ.DBGTOP_PWR_REQ to `0b1` and waiting for BSYS_PWR_ST.DBGTOP_PWR_ST to become `0b1`

  ——— **Note** ———

  This can be done in either the Host or Secure Enclave Base System Control registers.

  ———————————

- Using a software IMPLEMENTATION DEFINED method

——— **Note** ———

Arm recommends that:

- Debug agents using JTAG/SWD use the DP ROM **CDBGPWRUPREQ0/ACK0** method.
- All other debug agents use the BSYS_PWR_REQ and BSYS_PWR_ST registers, or the software IMPLEMENTATION DEFINED method.

———————————

The DBGTOP power domain can be extended by an implementation, using the Q-Channels of the DBGTOPQ interface. For more information on the DBGTOPQ interface, see *3.8.2 DBGTOP Q-Channel (DBGTOPQ) interface* on page 3-57.

5.4.4 CLUSTOP

The CLUSTOP power domain contains the following logic:

- Host CPU cluster logic and L2 cache: The Host CPU cluster logic includes other components. For more information, see the respective Technical Reference Manual for the Host CPU.
- Host GIC.

The CLUSTOP domain is controlled by CLUSTOP PPU and supports the following power modes:

- WARM_RST: All logic is powered, but the functional logic is held in reset. The debug logic within the Host CPU Cluster is available.
- ON: All logic and L2 cache RAMs are powered
- FUNC_RET: Enters L2 cache RAMs into retention. All other logic remains powered.

- MEM_RET: Enters L2 cache RAMs into retention. All other logic is not powered.
- OFF: All logic and L2 cache is not powered.

The CLUSTOP PPU is in the SYSTOP power domain. However, the PCSM associated with the PPU is in the AONTOP domain. The CLUSTOP PPU supports direct transitions from OFF to MEM_RET. This enables the PPU to be restored back to the power mode it was in before SYSTOP entered the OFF-power mode.

Note

WARM_RST can only be entered using static transitions. The software requesting these transitions must not be executing on the Host CPU. Otherwise, a deadlock occurs.

The following applies when the CLUSTOP domain is in the:

- OFF power mode:
  - An access on the HOSTCPUDBG interface gets an error response and does not cause a change in the power mode.
  - A CTI event on the HOSTCPUCTICHIN/OUT interfaces is ignored and does not cause a change in the power mode.
  - An access to the GICM interface is stalled and a request is generated to enter the FUNC_RET power mode.

    Note

    SSE-700 supports the L2 cache flush using the **L2FLUSHREQ** and **L2FLUSHACK** signals when CLUSTOP transition into the OFF power mode. See the *Arm® Cortex®-A32 Processor Technical Reference Manual* for more information on L2 Cache flush.

    Note

    The software must manually clean any dirty cache lines in the L2 for any locations that cannot be written using the same address that was used to fetch the initial data, for example, data initially fetched from a NAND flash device that has been modified.

- MEM_RET power mode:
  - An access on the HOSTCPUDBG interface generates an error response and does not cause a change in the power mode.
  - A CTI event on the HOSTCPUCTICHIN/OUT interfaces is ignored and does not cause a change in the power mode.
  - An access to the GICM interface is stalled and causes a request to enter the FUNC_RET power mode.
  - When CLUSTOP is exiting the MEM_RET mode the contents of the L2 cache is preserved in SSE-700.
- FUNC_RET power mode:
  - An access on the HOSTCPUDBG interface is allowed to the Host CPU external debug registers.
  - A CTI event on the HOSTCPUCTICHIN/OUT interface is allowed.
  - An access to the GICM interface is allowed.
  - Access on the HOSTCPUMEM interfaces is allowed.
- ON power mode:
  - An access on the HOSTCPUDBG interface is allowed to the Host CPU external debug registers.
  - A CTI event on the HOSTCPUCTICHIN/OUT interface is allowed.
  - Access to the GICM interface is allowed.
  - Access on the HOSTCPUMEM interfaces are allowed.
- WARM_RST power mode:

— An access on the HOSTCPUDBG interface generates an error response and does not cause a change in the power mode.
— A CTI event on the HOSTCPUCTICHIN/OUT interfaces is ignored and does not cause a change in the power mode.
— An access to the GICM interface is stalled and causes a request to enter the FUNC_RET power mode.

Warm_RST for CLUSTOP

WARM_RST power-mode is intended to be used for debugging only. It is not used during normal operation.

When CLUSTOP transitions into WARM_RST power mode, the logic within CORE{0-3} power domains are also reset.

———— **Note** ————

Arm strongly recommends that the CORE{0-3} PPUs are in OFF power mode when setting the policy to WARM_RST for the CLUSTOP PPU. SSE-700 does not support CLUSTOP transition into WARM_RST power mode if CORE{0-3} power domains are not in OFF mode.

————————————

5.4.5 CORE{0-3}

Each CORE{0-3} power domain contains a single Host CPU core.

The number of CORE power domains equals the number of cores in the Host CPU. Each CORE power domain is controlled by an associated CORE{0-3} PPU, which supports the following power modes:

- WARM_RST: Resets the functional logic in the CORE domain
- ON: All logic in the CORE domain is powered
- FULL_RET: All logic in the CORE domain is in a retention state
- OFF_EMU: All logic in the CORE domain is powered. However, the functional logic is held in reset.
- OFF: All logic in the CORE domain is not powered

The CORE PPU and PCSMs are in the SYSTOP power domain.

The following applies when the CORE{0-3} domain is in the:
- OFF power mode: Any access to the debug logic within the Host CPU core {x} generates an error response.
- OFF_EMU power mode: Any access to the debug logic within the Host CPU core {x} is allowed.
- FULL_RET power mode: Any access to the debug logic within the Host CPU core {x} is allowed.
- ON power mode: Any access to the debug logic within the Host CPU core {x} is allowed.
- WARM_RST power mode: Any access to the debug logic within the Host CPU core {x} is allowed.

WAKEUPREQ

When the **WAKEUPREQ[x]** signal is HIGH, there is a request to send an interrupt to the Host CPU core. The power control logic only responds to this signal in the OFF or OFF_EMU power modes. Otherwise it is ignored.

Core Primary Boot

Overview of the behavior of the CORE{0-3} power domains.

The power control logic for CORE{0-3} power domains enables the selection of the Host CPU cores that automatically enter the ON power mode when the CLUSTOP power domain performs one of the following transitions:

- OFF to FUNC_RET/ON/WARM_RST.
- MEM_RET to FUNC_RET/ON/WARM_RST.

By default, CORE0 is selected, but software can configure this using the HOST_CPU_BOOT_MSK register in the Host Base System Control.

Note

Arm recommends that only a single Host CPU core is enabled in the HOST_CPU_BOOT_MSK register at any given time.

5.4.6 SECENCTOP

Summary of the SECENCTOP power domain.

For details of the SECENCTOP power domain, see the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

5.4.7 External System Power Domains

Summary of the EXTSYS{0-1}TOP power domains.

Each External System has a power domain, referred to as EXTSYS{0-1}TOP. It is IMPLEMENTATION DEFINED, and includes the following logic:

- Sender frames of MHUs between the External and Host System. (ES{x}H{0-1}, where x is the External System number).
- Sender frames of MHUs between the External System and Secure Enclave. (ES{x}SE{0-1}, where x is the External System number).
- Receiver frames of MHUs between the Host and External System. (HES{x}{0-1}, where x is the External System number).
- Receiver frames of MHUs between the Secure Enclave and External System. (SEES{x}{0-1}, where x is the External System number).
- External System logic, which is IMPLEMENTATION DEFINED.
- Domain crossing logic for the following interfaces:
  - EXTSYS{0-1}MEM.
  - EXTSYS{0-1}MHU.
  - EXTSYS{0-1}TRACEEXP.
  - EXTSYS{0-1}DBG.
  - EXTSYS{0-1}EXTDBGAPB.
  - EXTSYS{0-1}CTICHIN.
  - EXTSYS{0-1}CTICHOUT.

An error is generated for an access to the External System {x} region of the External Debug Bus memory map. When the EXTSYS{x}TOP power domain is in a power mode where the EXTSYS{0-1}DBGQ interface is in the Q_STOPPED state.

For more information on the External Debug Bus memory map, see *11.1.2 External Debug Bus memory map* on page 11-169.

5.5 Power domain relationship

The relationships between specific power domains in the SSE-700 are:

- AONTOP
  - Relatively always-on to all other power domains in the SoC. This includes any other power domains added by the SoC implementor.
- SYSTOP
  - Relatively always-on to CLUSTOP. CLUSTOP must be in OFF or MEM_RET power mode before SYSTOP enters the OFF or MEM_RET power mode.
  - Independent relationship with DBGTOP.
  - Independent relationship with EXTSYS{0-1}TOP.
- DBGTOP
  - Independent relationship with CLUSTOP, SYSTOP, SECENCTOP and EXTSYS{0-1}TOP.
- SECENCTOP
  - Independent relationship with SYSTOP and DBGTOP.
  - Independent relationship with EXTSYS{0-1}TOP.
- CLUSTOP
  - Relatively always-on to CORE{0-3}.
    - When any CORE{0-3} is in any of the ON, FULL_RET or OFF_EMU power modes, then CLUSTOP must be in ON or FUNC_RET power mode.
    - When all CORE{0-3} are in OFF power mode, then CLUSTOP can enter MEM_RET or OFF.
- CORE{0-3}
  - Independent relationship between CORE{0-3} power domains.
- EXTSYS{0-1}TOP
  - Independent to SYSTOP, DBGTOP and between EXTSYS{0-1} power domains.
  - Independent relationship with SECENCTOP.

5.6 Power states

The power states represent a software-visible abstraction of available power modes of each power domain.

A power state defines the wakeup capability, loss of context, and the power consumption. This section describes power states for the SSE-700, but the External Systems can define their own power states.

The power state of SSE-700 is controlled using the following registers:

- Host Base System Control registers:
  - HOST_CPU_CLUS_PWR_REQ
  - BSYS_PWR_REQ
- Secure Enclave Base System Control registers:
  - BSYS_PWR_REQ

The power state determines the behavior for shared resources, for example the SYSTOP and REFCLK. Alongside the shared resources, there are implemented behaviors for the dedicated resources, such as the CLUSTOP power domain of the Host System.

No system has overall control of the power state and shared resources. The Host System and Secure Enclave both have copies of the BSYS_PWR_REQ register to request the power state. The External System does not have control of the power state directly but can make software requests to the Host System or Secure Enclave. All copies of the BSYS_PWR_REQ register, plus other hardware indicators are used to determine the power state of the SoC.

Table 5-1 SSE-700 power states

Power state	AON TOP	FW SR	SYS TOP	DBG TOP	CLUS TOP	SECEN CTOP	EXTSYS {0-1}TOP	REF CLK	S32K CLK	VSYS
BSYS.RUN	ON	ON/ FUNC _RET[a]	Any[b]	Any	Any[bcd]	Any	Any	Yes	Yes	ON
BSYS.SLEEP0	ON	ON/ FUNC _RET[a]	MEM_ RET /OFF	Any	MEM_ RET / OFF[cd]	Any	Any	Yes	Yes	ON
BSYS.SLEEP1	ON	ON/ FUNC _RET[a]	MEM_ RET /OFF	OFF	MEM_ RET / OFF[cd]	Any	Any	No	Yes	ON
BSYS.OFF	OFF	OFF[a]	OFF	OFF	OFF[cd]	OFF	OFF	No	No	OFF[e]

Note

Cells with a value of "Any" mean that the power domain can enter any legal power mode for that domain.

In BSYS.SLEEP1 where **REFCLK** is marked as OFF, it is legal for **REFCLK** to remain running and be used by other components outside SSE-700, within the SoC. However, the **REFCLK** is internally gated by SSE-700. Any component in the SSE-700 using **REFCLK** no longer has a clock. For example, the

[a] Host System Firewall RAM.
[b] SYSTOP and CLUSTOP cannot be in any of the following: SYSTOP and CLUSTOP in OFF; SYSTOP in OFF and CLUSTOP in MEM_RET; SYSTOP in MEM_RET and CLUSTOP in OFF; SYSTOP and CLUSTOP in MEM_RET.
[c] For the CLUSTOP to be in OFF or MEM_RET, all the CORE{x} power domains must be in OFF. The CORE{x} power domains can be in any legal power mode when CLUSTOP is in FUNC_RET or ON.
[d] Whenever the CLUSTOP domain is not in OFF the VCLUS, if implemented, must be ON.
[e] In BSYS.OFF, VSYS can be removed, but this is not required. Under some conditions, the system enters BSYS.OFF but VSYS is not removed, that is, **nSRST** is asserted.

REFCLK timestamp does not update and software must reprogram the time from the S32K counter on entry into BSYS.SLEEP0 or BSYS.RUN.

When transitioning between any power state, except all transitions to BSYS.OFF and BSYS.OFF-> BSYS.SLEEP1, **REFCLK** is required and is requested by SSE-700 using the REFCLKQ interface.

The following sections show the conditions required to transition to a power state.

Power state transition conditions for BSYS.RUN to BSYS.SLEEP0

When all the following are true:

- CLUSTOP power domain is in MEM_RET or OFF power mode.
- SYSTOP power domain is in MEM_RET or OFF power mode.
- Host or Secure Enclave Base System Control BSYS_PWR_REQ.REFCLK_REQ is `0b1`.
- Host and Secure Enclave Base System Control BSYS_PWR_REQ.SYSTOP_PWR_REQ is `0b000` or `0b001`.
- HOST_CPU_CLUS_PWR_REQ.PWR_REQ is `0b0`.
- When Host Base System Controller BSYS_PWR_REQ.WAKEUP_EN is 0b1, then the following conditions must be true:

  — GICWAKEUP is 0b0

  The following interrupts are not asserted:

  — Secure Watchdog WS0
  — Non-secure Watchdog WS0 or WS1
  — UART{0,1} UARTINTR.
  — HSE{0-1} Combined IR
  — SEH{0-1} Combined IRQ
  — HES{0-1}{0-1} Combined IRQ
  — ES{0-1}H{0-1} Combined IRQ
  — Host UART{0,1} UARTINTR

  The following interrupts are either not asserted or not routed to the Host GIC via the Interrupt Router:

  — REFCLK Timer {0-3} and S32KCLK Timer {0,1}
  — SDC-600
  — Host System Firewall
  — Host PPU Combined

  ———— **Note** ————

  When Host Base System Control BSYS_PWR_REQ.WAKEUP_EN is `0b0`, the following signals are considered to be `0b0`:

  — GICWAKEUP
  — Status of the following interrupts:
    ◦ Secure Watchdog WS0
    ◦ Non-secure Watchdog WS0 or WS1
    ◦ HSE{0-1} Combined IRQ
    ◦ SEH{0-1} Combined IRQ
    ◦ HES{0-3}{0-1} Combined IRQ
    ◦ ES{0-3}H{0-1} Combined IRQ
    ◦ Host UART{0,1} UARTINTR
  — Status of the following interrupts, independent of their routing via the Interrupt Router:
    ◦ REFCLK Timer {0-3}
    ◦ S32KCLK Timer {0,1}
    ◦ SDC-600
    ◦ Host System Firewall
    ◦ Host PPU Combined

  ————————————

- Host ROM **CDBGPWRUPREQ0** is LOW.
- AXIAP ROM **CSYSPWRUPREQ[1:0]** are both LOW.
- There are no outstanding transactions from/on any of the following masters or interfaces to the Host System address space:

  — Host ETR or CATU
  — SoC ETR or CATU
  — AXI-AP

  - External System {0-1}
  - Secure Enclave and GICM
- SDC-600 REMPUR is `0b0`.
- **CLUSTOPINGRESSQACTIVE** is LOW.
- **CLUSTOPEGRESSQACTIVE** is LOW.
- All SYSTOPQ interfaces **QACTIVE** is LOW.

Power state transition conditions for BSYS.SLEEP0 to BSYS.RUN

When any of the following occurs:

- **GICWAKEUP** is HIGH and Host Base System Control BSYS_PWR_REQ.WAKEUP_EN is `0b1`.
- Any of the following interrupts are asserted and the Host Base System Control BSYS_PWR_REQ.WAKEUP_EN is `0b1`:
  - Secure Watchdog WS0.
  - Non-secure Watchdog WS0 or WS1.
  - HSE{0-1} Combined IRQ.
  - SEH{0-1} Combined IRQ.
  - HES{0-1}{0-1} Combined IRQ.
  - ES{0-1}H{0-1} Combined IRQ.
  - UART{0,1} UARTINTR.
- Any of the following interrupts are asserted, routed to the Host GIC through the Interrupt Router and the Host Base System Control BSYS_PWR_REQ.WAKEUP_EN is `0b1`:
  - REFCLK Timer {0-3}.
  - S32KCLK Timer {0,1}.
  - SDC-600.
  - Host System Firewall.
  - Host PPU Combined.
- Host ROM **CDBGPWRUPREQ0** is HIGH.
- AXIAP ROM **CSYSPWRUPREQ0** is HIGH.
- AXIAP ROM **CSYSPWRUPREQ1** is HIGH.
- There is an outstanding transaction from/on any of the following masters or interfaces to the Host System address space:
  - Host ETR or CATU.
  - SoC ETR or CATU.
  - AXI-AP.
  - External System {0-1}.
  - Secure Enclave.
  - GICM.
- HOST_CPU_CLUS_PWR_REQ.PWR_REQ is `0b1`.
- Host Base System Control BSYS_PWR_REQ.SYSTOP_PWR_REQ field is `0b010` or greater.
- Secure Enclave Base System Control BSYS_PWR_REQ.SYSTOP_PWR_REQ field is `0b010` or greater.
- SDC-600 **REMPUR** is HIGH.
- Any of the following MHUs have ACCESS_REQUEST.ACC_REQ set to `0b1`:
  - SEH{0,1}
  - ES{0-1}H{0,1}
- Any SYSTOPQ interface's QACTIVE is `0b1`

Power state transition conditions for BSYS.SLEEP0 to BSYS.SLEEP1

When all the following conditions are true:

- All BSYS_PWR_REQ.SYSTOP_PWR_REQ are set to `0b00x`.
- CLUSTOP power domain is in MEM_RET or OFF power mode.
- SYSTOP power domain is in MEM_RET or OFF power mode.
- Both BSYS_PWR_REQ.REFCLK is set to `0b0`.
- DBGTOP power domain is in OFF power mode.
- DP **CDBGPWRUPREQ** is LOW.
- **GICWAKEUP** is LOW or Host Base System Control BSYS_PWR_REQ.WAKEUP_EN is 0b0.
- All of the following interrupts are not asserted or the Host Base System Control BSYS_PWR_REQ.WAKEUP_EN is `0b0`:
  - Secure Watchdog WS0.
  - Non-secure Watchdog WS1.
  - Non-secure Watchdog WS0.
  - HSE{0-1} Combined IRQ.
  - SEH{0,1} Combined IRQ.
  - HES{0-1}{0,1} Combined IRQ.
  - ES{0-1}H{0-1} Combined IRQ.
  - UART{0,1} UARTINTR.
- All of the following interrupts are not asserted or are not routed to the Host GIC using the Interrupt Router or the Host Base System Control BSYS_PWR_REQ.WAKEUP_EN is `0b0`:
  - REFCLK Timer {0-3}.
  - S32K Timer {0,1}.
  - Host System Firewall.
  - Host PPU Combined.
  - SDC-600.
- No memory access from the Secure Enclave or any External Systems {0-1} to the Host System address space.
- Host or Secure Enclave BSYS_PWR_REQ.SYSTOP_PWR_REQ is `0b000` or `0b001`.
- All of the following MHUs have ACCESS_REQUEST.ACC_REQ set to `0b0`:
  - SEH{0,1}.
  - ES{0-1}H{0,1}.
- All SYSTOPQ interface **QACTIVE** is LOW.

Power state transition conditions for BSYS.SLEEP1 to BSYS.RUN

When any of the following occurs:

- **GICWAKEUP** is HIGH and Host Base System Control BSYS_PWR_REQ.WAKEUP_EN is `0b1`.
- Any of the following interrupts are asserted and the Host Base System Control BSYS_PWR_REQ.WAKEUP_EN is `0b1`:
  - Secure Watchdog WS0.
  - Non-secure Watchdog WS1.
  - Non-secure Watchdog WS0.
  - HSE{0-1} Combined IRQ.
  - SEH{0,1} Combined IRQ.
  - HES{0-1}{0-1} Combined IRQ.
  - ES{0-1}H{0-1} Combined IRQ.
  - Host UART{0,1} UARTINTR.
- Any of the following interrupts are asserted, the Host Base System Control BSYS_PWR_REQ.WAKEUP_EN is `0b1` and the interrupt is routed to the Host GIC by the Interrupt Router:
  - REFCLK Timer {0-3}.
  - S32K Timer {0,1}.
  - Host System Firewall.
  - Host PPU Combined.
  - SDC-600.
- A memory access from the Secure Enclave or External Systems {0-1}.
- SDC-600 REMPUR is `0b1`
- Host or Secure Enclave BSYS_PWR_REQ.SYSTOP_PWR_REQ is `0b010` or greater.
- Any of the following MHUs have ACCESS_REQUEST.ACC_REQ set to `0b1`:
  - SEH{0,1}.
  - ES{0-1}H{0,1}.
- Any SYSTOPQ interface QACTIVE is `0b1`.

Note

Wakeup sources which are from components which use the **REFCLK** can only generate wakeup events if the event occurred before the SSE-700 entered the SLEEP1 state. Software must not rely on these sources for wakeup from SLEEP1 and is expected to move any wakuep sources to components which have a clock or select a clock source for the clock which is running in that power state.

Power state transition conditions for BSYS.SLEEP1 to BSYS.SLEEP0

When any of the following occur:

- DP **CDBGPWRUPREQ** is HIGH.
- Host or Secure Enclave Base System Control BSYS_PWR_REQ.DBGTOP_PWR_REQ field becomes `0b1`.
- Host or Secure Enclave Base System Control BSYS_PWR_REQ.REFCLK_REQ field becomes `0b1`.

Power state transition conditions for BSYS.OFF to BSYS.SLEEP1

VSYS is applied and **PORESETn** is deasserted.

Note

The Power on Reset sequence causes SSE-700 to perform the following sequence: BSYS.OFF -> BSYS.SLEEP1 -> BSYS.SLEEP0.

Power state transition conditions for BSYS.{RUN/ SLEEP0/ SLEEP1} to BSYS.OFF

Any of the following occur:

- SSE-700 **PORESETn** is asserted.
- SSE-700 **nSRST** input is asserted.
- Secure Enclave Watchdog reset request is asserted.
- SoC Watchdog reset request is asserted.
- DP ROM **CSYSRSTREQ** is HIGH.
- DP **CDBGRSTREQ** is HIGH.
- VSYS is removed.

Note

Transition from BSYS.SLEEP1 to BSYS.RUN enters BSYS.SLEEP0 first, followed by a BSYS.SLEEP0 to BSYS.RUN transition.

When performing transitions from BSYS.RUN to BSYS.SLEEP0, when a change in the conditions during the transitions occurs, the transition completes first. Then, the conditions are re-evaluated and SSE-700 returns to BSYS.RUN. When performing transitions from BSYS.SLEEP0 to BSYS.SLEEP1, when a change in the conditions during the transitions occurs, whether or not the transition is completed depends on the timing of the arrival of the condition change:

- If a REFCLK request happens at the same time as REFCLKQ moves into Q_STOPPED, then SSE-700 enters into BSYS.SLEEP1.
- If a REFCLK request happens when REFCLKQ is in Q_REQUEST, then this request is denied, SSE-700 aborts the transition and returns to BSYS.SLEEP0.

This section contains the following subsections:

- *5.6.1 Secure Enclave sleep states* on page 5-89.
- *5.6.2 External System power states* on page 5-89.

5.6.1 Secure Enclave sleep states

The Secure Enclave supports the following sleep states:

- SLEEPING
- SLEEPDEEP
- SLEEPDEEP PG

For details of Secure Enclave Sleep States, see the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

5.6.2 External System power states

The power states of the External System are limited to the power domain EXTSYS{0-1}TOP and any other IMPLEMENTATION DEFINED power domains which are part of the External System.

The power states of the External System and the method by which a power state is selected is IMPLEMENTATION DEFINED as long as the following requirements are met:

- EXTSYS.OFF power state:
  - All power domains, within the External System are OFF.
  - The only exit condition from this state is via a power on reset of the External System.
  - If VEXT is implemented, then it can be turned OFF in this power state but must be able to be requested without software intervention.
- EXTSYS.RUN power state:
  - Must be able to perform its primary task, i.e. execute instructions if the **EXTSYS{0-1}CPUWAIT** signal is not asserted.
  - Debugger must be able to access the resources of the External System.
  - If VEXT is implemented, then it must be turned ON in this power state.

- Must be able to transition from EXTSYS.OFF to EXTSYS.RUN directly.
  - Part of the power on reset sequence for the External System.
- Must be able to transition from EXTSYS.RUN to EXTSYS.OFF directly.
  - In a controlled manner, using an IMPLEMENTATION DEFINED method. This is intended for a clean shutdown of the SSE-700.
  - Through any of the following:
    - **PORESETn** is asserted.
    - **nSRST** input is asserted.
    - Secure Enclave Watchdog reset request is asserted.
    - Secure Enclave Host System reset request is asserted.
    - DP ROM **CSYSRSTREQ** is asserted.
    - DP **CDBGRSTREQ** is asserted.
    - Removal of VSYS.
    - Removal of VEXT, if implemented.

Other power states can be implemented, between OFF and RUN, specific to the External System. For example, an EXTSYS.SLEEP can be defined where the External System enters a low-power state, with only minimal wakeup logic remaining powered, whilst the rest of the External System is not powered.

Chapter 6
Reset

This chapter describes the reset requests, inputs, and outputs of the SSE-700.

It contains the following sections:

6.1 Reset overview

The SSE-700 reset strategy is built around a reset tree that incorporates a Reset Controller and the PPUs.

The Reset Controller handles top-level reset conditions, for example **PORSETn** or debug ROM table resets. The PPUs handle reset of the power domain that they control.

The following figure shows the SSE-700 reset tree.

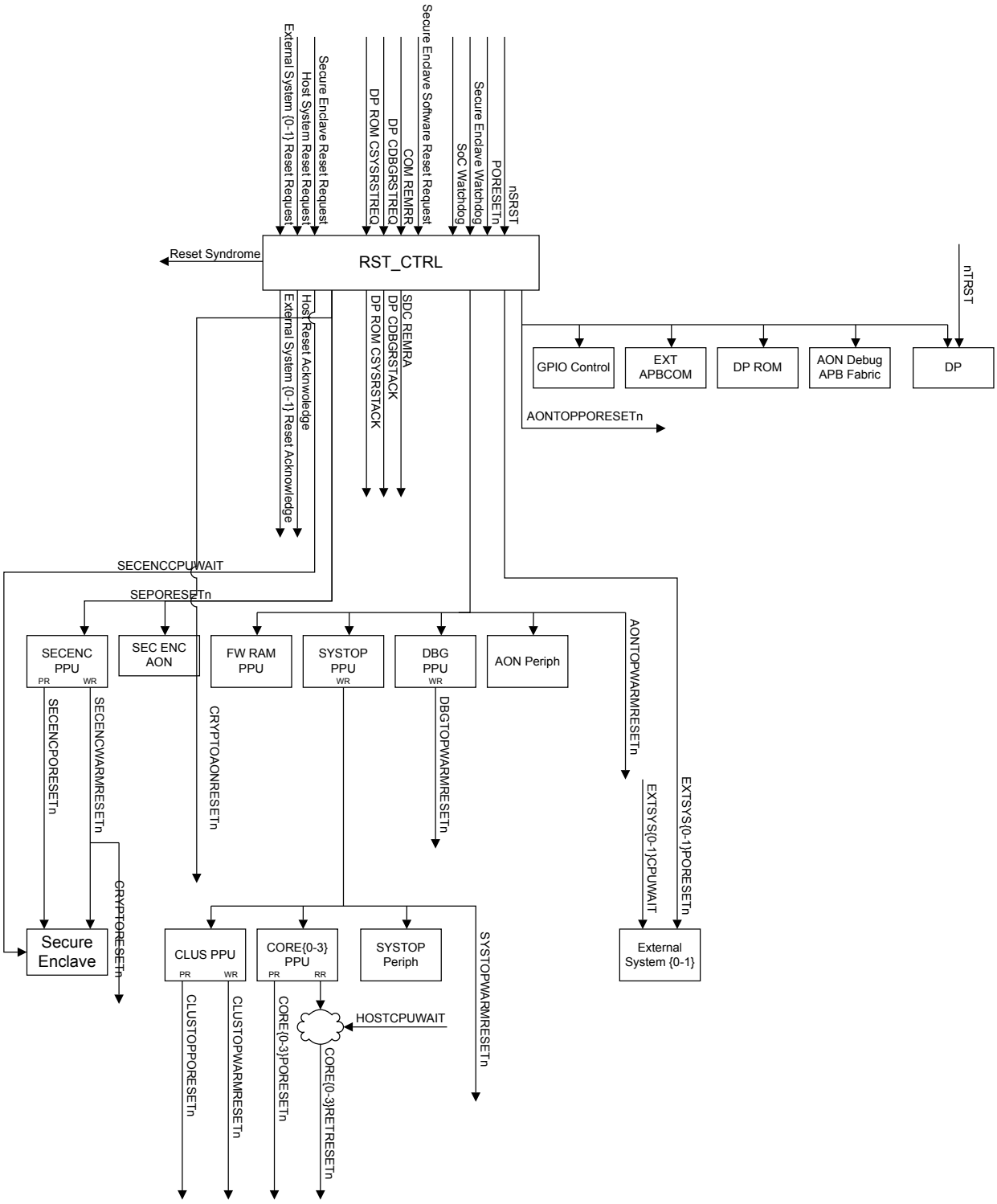

Figure 6-1 Reset tree

6.2 Reset inputs

All reset inputs to the SSE-700 subsystem are treated as asynchronous assert.

The **PORESETn** and **nTRST** resets are treated as asynchronous deassert, as the SSE-700 implements an internal reset synchronizer.

SSE-700 has the following reset inputs:

- **PORESETn**: SoC Power-on reset
- **nTRST**: Reset for the JTAG/Serial Wire Debug interface
- **EXTSYS{0-1}ARESETn**: Reset for the EXTSYS{0-1}MEM interface
- **EXTSYS{0-1}MHURESETn**: Reset for the EXTSYS{0-1}MHU interface
- **EXTSYS{0-1}DBGPRESETMn**: Reset for the EXTSYS{0-1}DBG interface
- **EXTSYS{0-1}DBGPRESETSn**: Reset for the EXTSYS{0-1}EXTDBG interface
- **EXTSYS{0-1}ATRESETn**: Reset for the EXSYS{0-1}TRACEEXP interface
- **EXTSYS{0-1}CTIRESETn**: Reset for the EXTSYS{0-1}CTICHIN and EXTSYS{0-1}CTICHOUT interfaces

6.3 Reset requests

The following table shows how the reset requests in SSE-700 provide different levels of reset.

Table 6-1 Reset Requests

Name	Type	Description
PORESETn	SSE-700 input	SoC Power-on reset. Top-level asynchronously asserted and deasserted active-LOW reset of SSE-700. The **PORESETn** input is synchronized within SSE-700 to the **S32KCLK** before being used, to reset the Reset Controller. When asserted, all logic in the SoC is reset. ——— **Note** ——— **PORESETn** is used to reset the JTAG/SWD interface of the Debug Port, however no synchronization of **PORESETn** takes place for the Debug Port. ———
nTRST	SSE-700 input	JTAG/SWD interface reset. Top-level asynchronously asserted and deasserted active-LOW reset of SSE-700. The **nTRST** input is used to reset the JTAG/SWD interface of the Debug Port.
nSRST	SSE-700 input	SoC Warm Halted Reset. Top-level asynchronously assert and deasserted active-LOW reset request. The **nSRST** input signal is internally synchronized to the **S32KCLK**. When the input is asserted, all logic in the SoC is reset, except for the logic on the **AONTOPPORESETn**. After some cycles, which are configured through the SOC_RST_DLY parameter, the SoC is released from reset. However, the Secure Enclave Cortex-M0+ is prevented from executing instructions while the **nSRST** signal is asserted LOW. This enables a debug agent to perform any initial setup of the debug logic of the SoC to facilitate debug from reset.
DP **CDBGRSTREQ**	SSE-700 internal	SoC Power-on reset request, from the Debug Port. Equivalent to a **PORESETn**, except the Reset Controller is not reset.
DP ROM **CDBGRSTREQ**	SSE-700 internal	DBGTOP reset. Debug reset request from the DP ROM. Causes a reset of the DBGTOP power domain.
DP ROM **CSYSRSTREQ**	SSE-700 internal	SoC Warm-halted reset. System reset request from DP ROM. Equivalent to **nSRST**. While **CSYSRSTREQ** is HIGH, the Secure Enclave Cortex-M0+ is prevented from executing instructions.
Secure Enclave Watchdog Reset Request	SSE-700 internal	SoC Power-on reset Reset request from the Secure Enclave Watchdog. Equivalent to PORESETn, except the Reset Controller is not reset. For more information on the use of the Secure Enclave Watchdog, see *12.5 Watchdog usage on page 12-248*.

Table 6-1 Reset Requests (continued)

Name	Type	Description
Secure Enclave Software Reset Request	SSE-700 internal	SoC Warm reset. Reset request from the Secure Enclave Cortex-M0+ driven by the AIRCR.SYSRESETREQ field. This reset is the equivalent to **nSRST**, except that the Secure EnclaveCortex-M0+ is not prevented from executing instructions after the reset has been applied. For more information, see the *Arm®v6-M Architecture Reference Manual*.
SoC Watchdog Reset Request	SSE-700 internal	SoC Power-on reset Reset request from the SoC Watchdog. Equivalent to **PORESETn**, except the Reset Controller does not reset. For more information on the usage of the SoC Watchdog, see *12.5 Watchdog usage on page 12-248*.
Power domain PPU	SSE-700 internal	Power domain reset. As the PPU enter the different power modes, resets are applied to various parts of the logic. The WARM_RST power mode can be used on any power domain to reset certain logic within the domain. Typically, this causes the functional logic to be reset, while the debug logic is not reset. Arm strongly recommends: • Software only sets the policy to WARM_RST when the domain is known to be idle. • Takes care when using WARM_RST to not enter a deadlock condition, caused by software being unable to update the policy once in WARM_RST.
Software External System Reset Request	SSE-700 internal	In the Host Base System Control registers there is a register for each External System that is implemented to enable a software request to reset the associated External System. When software sets the EXT_SYS{0-1}_RST_CTRL.RST_REQ bit to `0b1`, the **EXTSYS{0-1}PORESETn** is asserted. This causes all logic within the External System to be reset. **Note** It is not guaranteed that **EXTSYS{0-1}PORESETn** is asserted if the interfaces of the External System are not quiescent.
Software Host System Reset Request	SSE-700 internal	In the Secure Enclave Base System Control registers there is a register that enables software to request that the Host and all External Systems implemented are reset. When software sets the HOST_SYS_RST_CTRL.RST_REQ bit to `0b1`, the **AONTOPWARMRESETn** and **EXTSYS{0-1}PORESETn** are asserted. This causes all logic within the Host and External Systems to be reset. This also includes the debug logic of SSE-700, except for the External Debug Bus. **Note** It is not guaranteed that the **AONTOPWARMRESETn** is asserted if the interfaces between the Secure Enclave and the Host System are not quiescent.

Table 6-1 Reset Requests (continued)

Name	Type	Description
SoC Reset Request	SSE-700 internal	In the Secure Enclave Base System Control registers there is a register that enables software to request that the entire SoC is reset. When software sets the SOC_RST_CTRL.RST_REQ bit to `0b1`, all logic within the SoC is reset. This is equivalent to **PORESETn** being asserted, except that the Reset Controller is not reset.
DBGRSTREQ	Host CPU_GIC Socket input	Host CPU core reset. Request to reset the functional logic of the Host CPU core. Arm strongly recommends that this reset is only used when the Host CPU core is known to be idle. Requesting a reset when the Host CPU core is not idle can lead to unpredictable results.
WARMRSTREQ	Host CPU_GIC Socket input	Host CPU core reset. Request to reset the function logic of the Host CPU core. Software must execute a WFI before the reset of the Host CPU core is applied.
External System Reset Inputs	External System Harness inputs	The External System Harness defines the following reset input signals, which reset interfaces of the External System: • **EXTSYS{0-1}ARESETn** • **EXTSYS{0-1}MHURESETn** • **EXTSYS{0-1}DBGPRESETMn** • **EXTSYS{0-1}DBGPRESETSn** • **EXTSYS{0-1}ATRESETn** • **EXTSYS{0-1}CTIRESETn** **Note** These resets are generated by power control logic of the External System and must be either directly or indirectly asserted when **EXTSYS{0-1}PORESETn** is asserted.

Note

The difference between a SoC Warm reset and a SoC Warm-halted reset is as follows: during a SoC Warm-halted reset, the SoC is prevented from executing any instructions, while the conditions that caused it, or any other condition which can cause it, remain asserted.

6.4 Reset outputs

SSE-700 has several reset outputs. Each reset output hasr a specific function.

All reset outputs are asynchronously asserted and asynchronously deasserted. The resets are driven either:

- Directly from the **DEVPORESETn**, **DEVRETRESETn**, and **DEVWARMRESETn** outputs of the PPU
- From the Reset Controller

Note

Arm strongly recommends that reset outputs are only used as described in the following table.

Table 6-2 SSE-700 reset outputs

Reset Output	Usage
AONTOPPORESETn	Resets extended logic implemented by parterns within the AONTOP power domain, which is not affected by the **nSRST** functionality
AONTOPWARMRESETn	Resets extended logic implemented by partners within the AONTOP power domain, which is affected by the **nSRST** functionality
CRYTPORESETn	Resets the Crypto Accelerator
CRYPTOAONRESETn	Resets the Crypto Accelerator Always-on
SYSTOPWARMRESETn	Resets any extended logic implemented by partners within the SYSTOP power domain
DBGTOPWARMRESETn	Resets any logic within the DBGTOP power domain
EXTSYS{0-1}PORESETn	Either directly or indirectly resets all logic within the External System, including reset control logic of the External System. An External System reset only reset logic on the **EXTSYS{0-1}PORESETn**. An External System reset is caused by software setting the EXT_SYS{0-1}_RST_CTRL.RST_REQ field to `0b1`. The logic within the External System can be reset using an internal request, for example a watchdog, but the interfaces with the logic outside the External System must be quiescent before performing the reset.
HOSTCLUSTOPPORESETn	Resets logic, within the CLUSTOP power domain, which must not be affected by a WARM_RST
HOSTCLUSTOPWARMRESETn	Resets logic, within the CLUSTOP power domain, which must be affected by a WARM_RST
HOSTCPUPORESETn[x:0]	Resets debug logic of the Host CPU Core{0-3}. **Note** This is a superset reset of **HOSTCPUWARMRESETn[x:0]**.
HOSTCPUWARMRESETn[x:0]	Resets functional logic of the Host CPU Core{0-3}

Note

For **HOSTCPUPORESETn** and **HOSTCPUWARMRESETn**, x is equal to `HOST_CPU_NUM_CORES`-1.

6.5 Power-on reset

When the **PORESETn** signal is asserted, SSE-700 is in a *Power on Reset* (PoR) state.

During a PoR state, the **REFCLKQACTIVE** is asserted. To exit the PoR state, the following conditions must be true before the **PORESETn** signal is deasserted:

- **S32KCLK** must be running.
- **SECENCREFCLK** must be running.
- VSYS voltage supply must be available and stable.
- If the VCLUS voltage supply is implemented, then it must either be:
  - Available and stable
  - Requestable without the use of software running on any processor in the SoC when a debugger connects to the SoC through the SWJ interface. The method by which this is achieved is IMPLEMENTATION DEFINED.
- If VEXTSYS{0-1} voltage supplies are implemented, then they must either be:
  - Available and stable
  - Requestable without the use of software running on any processor in the SoC when a debugger connects to the SoC through the SWJ interface. The method by which this is achieved is IMPLEMENTATION DEFINED.

When exiting PoR state, SSE-700 automatically performs the following sequence:

- Transition to BSYS.SLEEP1 on deasserted of **PORESETn**. The SECENCTOP power domain automatically enters the ON power mode and the Secure Enclave Cortex-M0+ starts to execute instructions.
- When the REFCLK Q-Channel enters the Q_RUN state, SSE-700 transitions to BSYS.SLEEP0.

SSE-700 only enters the BSYS.RUN power state when a request for SYSTOP to exit the OFF power mode is detected.

Chapter 7
Debug

This chapter describes the SSE-700 debug infrastructure.

It contains the following sections:

7.1 Debug overview

SSE-700 provides a debug infrastructure for the Secure Enclave and Host System and provides interfaces for the debug infrastructure of the External Systems. It is compliant with the *Arm® Debug Interface Architecture Specification ADIv6.0*.

SSE-700 debug infrastructure provides the following features:

- Self-hosted debug of Host or External Systems
- External debug of Secure Enclave, Host, or External System by an off-chip debugger
- External debug of Secure Enclave, Host, or External System from one of the other systems in the SSE-700
- Full cross trigger support
- Trace support
- Granular power control
- Debug through power down, for the Host and External Systems
- Debug from reset, for all systems
- Fine grain control of debug privileges provided to the debug agent
- Support for single or multi-system debug
- Certificate injection using SDC-600
- Support for Serial Wire Debug
- Support for JTAG debug

In this document the following terms are used:

Target system
The system which is being debugged. The target system can be an individual system or multiple systems within the SSE-700. This variability is indicated by the term target system(s).

Debug agent
The entity performing the debug on the target system. This can be one of:
- Software running on the target system, for example, software running on a Host CPU core debugging itself or another Host CPU core in the Host System.
- Software running on another system within SSE-700. For example, an application on the Host CPU debugging an External System. This also includes where the software is running a USB or WiFi stack to perform debug over functional IO.
- Off-chip debugger, for example, a JTAG or Serial Wire debugger (SWJ).

Self-host debug agent
The debug agent is software running on a processor which is being debugged. For example, this is where a Host CPU core uses the System registers to access its own debug logic.

External debug agent
The debug agent is one of the following:
- Software running on another processor to the one being debugged. This includes when the processors are operating in a multiple processor configuration, all running the same operating system.
- Software running on another system.
- Off-chip debugger.

The debug infrastructure in SSE-700 is constructed from multiple separate blocks:
- External Debug Bus
- Host System Debug
- Secure Enclave Debug
- SoC Debug
- AXI AP Debug
- External System Debug

Some of the blocks provide debug features to a single system, whilst others provide features to all systems within SSE-700.

7.2 Authentication

SSE-700 uses *Debug Authentication Zones* (DAZ) and *Debug Authorization Access Control Gate* (DAACG) to:

- Control the types of debug:
  - Non-secure/secure
  - Invasive/non-invasive
- Control which debug agents can debug a specific target system.:
  - Self-hosted
  - External: It can control which debug agents can be the external debuggers for a target system

This section contains the following subsections:

- *7.2.1 Debug Authentication Zone (DAZ)* on page 7-102.
- *7.2.2 Debug Authorization Access Control Gate (DAACG)* on page 7-103.

7.2.1 Debug Authentication Zone (DAZ)

DAZ controls which debug features are enabled for the debug logic within the zone.

A DAZ is controlled by a unique set of the Arm CoreSight authentication signals: **DBGEN**, **NIDEN**, **SPIDEN**, and **SPNIDEN**. The values of the authentication signals are driven by the SCB of the Secure Enclave. Not all DAZ support all the authentication signals. The following table describes each DAZ in the SSE-700.

Table 7-1 DAZ Signals

DAZ name	Components	Extendable	DBGEN	NIDEN	SPIDEN	SPNIDEN
DPAUTH	DP ROM EXTDBG ROM	No	Y	Y	Y	Y
COMAUTH	SDC-600	No	Y[a]	Y[a]	N	N
HOSTAXIAUTH	Host AXI AP Host AXIAP ROM	No	Y	Y	Y	Y
HOSTEXTAUTH [b]	Host APB AP	No	Y	Y	Y	Y
HOSTAUTH	Host ROM Host ETR Host STM Host CTI Host Replicator Host CPU Cluster [c] Host Expansion Host CATU	Yes	Y	Y	Y	Y

[a] For the COMAUTH, the **DBGEN** signal is referred to as the PEN and NIDEN is referred to as the RRDIS. These signals then drive the **CFG_PEN** and **CFG_RRDIS** inputs of EXT APBCOM.

[b] There is no difference between invasive and non-invasive debug and therefore:
- **DBGEN** and **NIDEN** and referred to as NS and set to the same value.
- **SPIDEN** and **SPNIDEN** and referred to as S and set to the same value.

For example, the HOSTEXTAUTH DAZ has two signals: NS and S. The NS signal sets the value of **DBGEN** and **NIDEN** of the Host APB AP. The S signal sets the value of **SPIDEN** and **SPNIDEN** of the Host APB AP.

[c] The Host CPU Cluster includes a number of debug components. For more information, see the *Arm® Cortex®-A32 Processor Technical Reference Manual*.

Table 7-1 DAZ Signals (continued)

DAZ name	Components	Extendable	DBGEN	NIDEN	SPIDEN	SPNIDEN
TPIUAUTH	SoC TPIU SoC TPIU Funnel SoC TPIU Replicator SoC ETR SoC CATU SoC CTI	No	Y	Y	Y	Y
COUNTERAUTH	Counter CTI	No	Y	Y	Y	Y
SECENCAUTH [d]	Secure Enclave AHB AP[e] Secure Enclave CTI Secure Enclave Cortex-M0+ [f] Secure Enclave CS ROM	No	Y	Y	N	N

There are two types of DAZ:

- Controls the level of debug within a system, for example HOSTAUTH
- Controls of external debug access to the system, for example HOSTEXTAUTH

7.2.2 Debug Authorization Access Control Gate (DAACG)

The *Debug Authorization Access Control Gate* (DAACG) is a type of *Access Control Gate* (ACG) which grants access from master(s) to downstream components.

ACG status is controlled by the SCB of the Secure Enclave. The ACG is either in:

Open state
DBGEN is HIGH. Accesses are allowed through the ACG.

Closed state
DBGEN is LOW. Accesses are terminated by the DAACG with an error.

SSE-700 includes an instance of the DAACG for each master interface to the External Debug Bus. Using the DAACGs enables the software to configure which external debug agents are able to access the components on the External Debug Bus.

Note

Arm strongly recommends that when **DBGEN** is changed, no transactions are outstanding on the slave interface. It is UNPREDICTABLE whether the transaction sees the new or old value of **DBGEN**.

You can enable debug from more than one external debug agent. However, you cannot grant different external debug agents different debug privileges to the same target system or to different target systems. Arm strongly recommends that only a single external debug agent is enabled at any time.

You can have one target system performing self-hosted debug while another system is being debugged by an external debug agent. This also includes where the debug agent is debugging another CPU within the same system, for example, one Host CPU core debugging another Host CPU core.

[d] Secure Enclave only has a single security world and any debug components which support both secure and non-secure debug have the same debug privileges enabled in either security world. This is achieved by driving the **DBGEN** and **SPIDEN** signals and the **NIDEN** and **SPNIDEN** signals to the same value.
[e] The ap_en and ap_secure_en are both driven by the logical OR of the **DBGEN** and **NIDEN** from the SECENCAUTH DAZ.
[f] The Secure Enclave Cortex-M0+ includes a number of debug components. For more information, see *Cortex®-M0+ Technical Reference Manual*.

7.3 Debug blocks

The following sections describe the blocks that form the debug infrastructure of the SSE-700.

This section contains the following subsections:

7.3.1 External Debug Bus

The following figure shows the External Debug Bus, which provides access to any debug agent that is performing external debug on a target system:

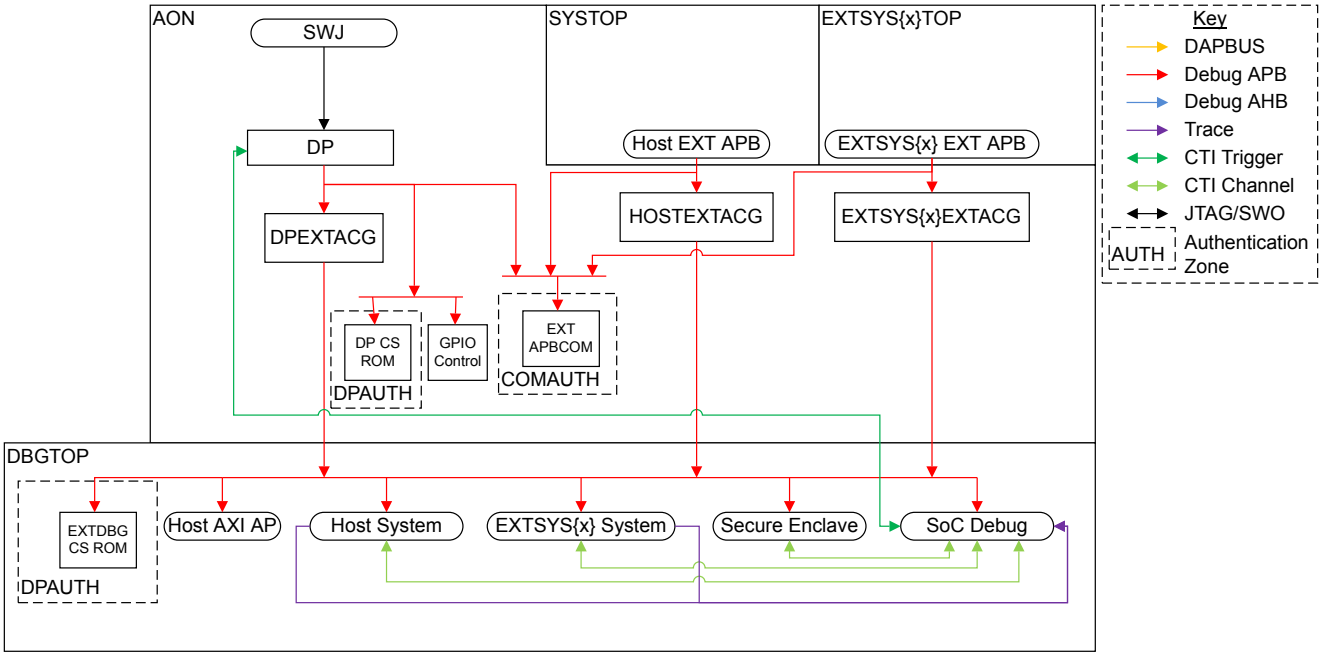

Figure 7-1 External Debug Bus

Note

This figure shows a single EXTSYS{x} EXT APB interface and EXTSYS{x} system. There is one for each External System in SSE-700.

The External Debug Bus contains the following:

- *Debug Port* (DP): Provides access to an off-chip JTAG or a Serial Wire Debug Agent
- Host External APB (Host EXT APB): Provides access for the Host System to be an external debug agent to any system in the SSE-700
- External System {0-1} External APB (EXTSYS{x} EXT APB): Provides access for the External System {0-1} to be an external debug agent to any system in the SSE-700
- EXT APBCOM: Provides the ability to insert a certificate to enable debug access for an external debug agent

Note

When the debug agent is software running on a system within the SoC, Arm strongly recommends an IMPLEMENTATION DEFINED method is used to request access to the External Debug Bus, instead of

using the EXT APBCOM. For more information on the EXT APBCOM, see *9.7 CoreSight SDC-600* on page 9-155.

- Debug Port ROM (DP ROM).
  - ROM table for DP, pointing to EXT APBCOM, EXTDBG ROM and GPIO Control.
- External Debug ROM (EXTDBG ROM)
  - ROM table for all components on the External Debug Bus, except for EXT APBCOM and GPIO Control.
- GPIO Control (GPIO Control)
  - Debug Component to be able to control general purpose inputs and outputs. In SSE-700 the GPIO Control component supports a single GPO and no GPI. The GPO is used to drive the CALC interface, along with the SOCLCC interface. For more information see *7.10 GPIO control* on page 7-123.

    Note

    The GPIO Control block has its **DBGEN** input tied HIGH in SSE-700.

- Debug APB ports to:
  - Host System Debug (Host System)
  - Host System (Host AXI AP)
  - External System Debug (EXTSYS{x} System)
  - Secure Enclave Debug (Secure Enclave)
  - SoC Debug logic (SoC Debug)
- Two DAZ:
  - DPAUTH: Controls the level of debug functionality of the DP and EXTDBG ROM.
  - COMAUTH: Controls the level of functionality provided by the EXT APBCOM. **DBGEN** drives the **CFG_PEN** input and **NIDEN** drives the **CFG_RRDIS** input, through an inverter.
- The following DAACGs:
  - DPEXTACG: Prevents access to all debug logic other than the DP ROM, EXT APBCOM and GPIO Control from the DP.
  - HOSTEXTACG: Prevents access to all debug logic other than EXT APBCOM, from the Host System using the Host EXT APB interface.
  - EXTSYS{x}EXTACG: Prevents access to all debug logic other than EXT APBCOM, from the External System using the EXTSYS{x}EXTAPB interface.

11.1.2 External Debug Bus memory map on page 11-169 shows the memory map for the External Debug Bus.

7.3.2 Host AXI AP debug

The AXI AP and the associated AXI ROM table provide direct access into the Host System memory map, as the following figure shows.

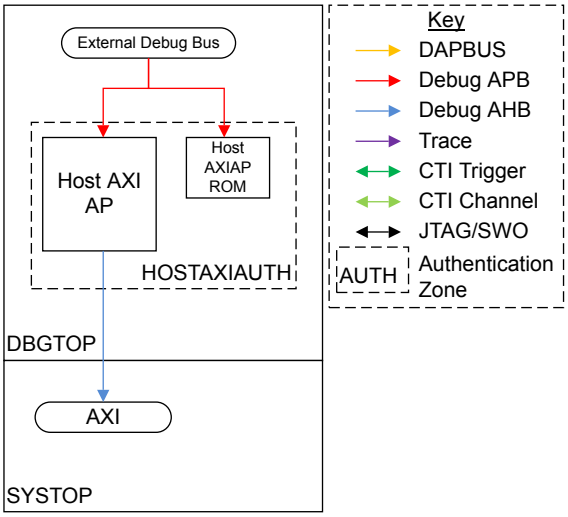

Figure 7-2 AXI AP

For information on the Host System memory map, see *11.1.1 Host System memory map* on page 11-163. Both the AXI AP and ROM are part of the HOSTAXIAUTH authentication zone. The AXI AP ROM provides power control for the power domains of the SoC which can be accessed via the Host AXI AP. For more information, see*7.7 ROM tables* on page 7-117.

7.3.3 Secure Enclave debug

The Secure Enclave provides access to the Secure Enclave debug logic through an AHB AP.

The Secure Enclave debug logic provides the following:

- Secure Enclave AHB AP
- Secure Enclave CS ROM
- Secure Enclave CTI
- Secure Enclave Channel Gate
- Secure Enclave Cortex-M0+ Debug

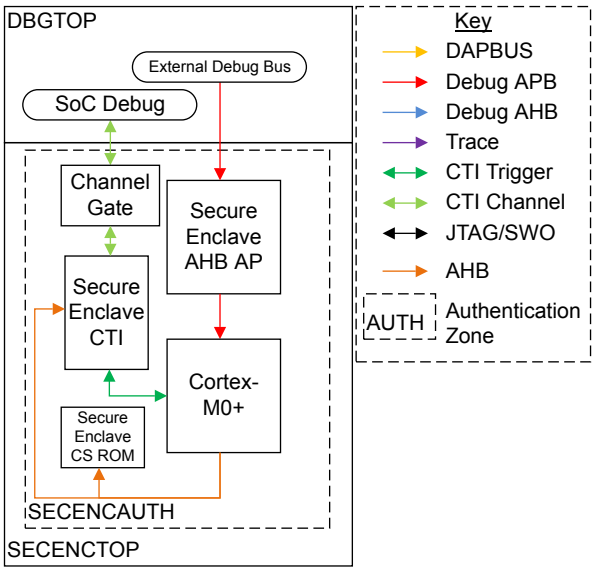

Figure 7-3 Secure Enclave Debug Access

The Secure Enclave AHB AP, and all debug logic within the Secure Enclave, is part of the SECENCAUTH authentication zone.

7.3.4 Host System debug

The following figure shows the Host System debug block.

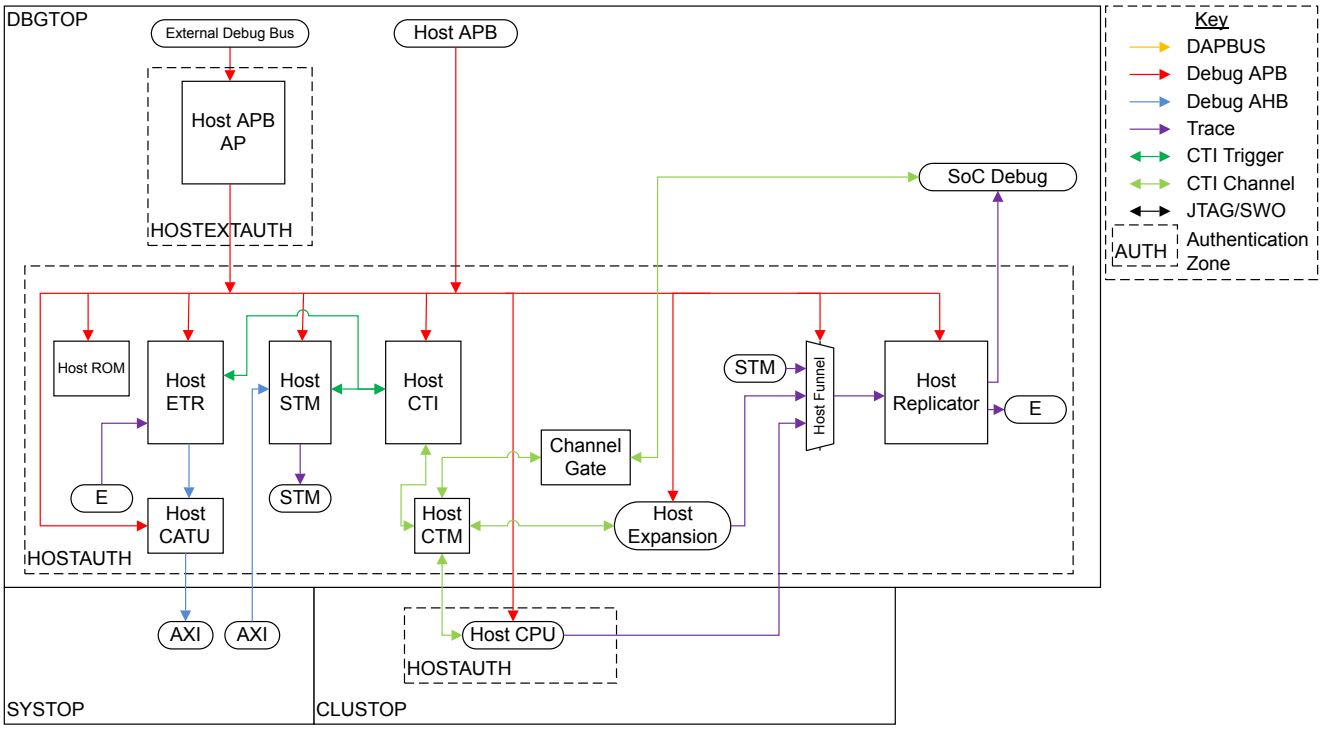

Figure 7-4 Host System debug block

The Host System debug block provides the following features:

- Host APB AP: Provides access for an external debug agent
- Host ROM: CoreSight ROM table with GPR functionality
- Host ETR: Ability to send only Host System trace to memory
- Host CATU: Ability to perform address translation for the Host ETR AXI transactions
- Host STM: Ability to perform instrumented software trace
- Host CTI: Trigger capabilities for Host ETR and STM
- Host CTM and Channel Gate
- Host Funnel: Combines trace from Host STM, HOSTCPUTRACE, and HOSTTRACEEXP interfaces
- Host Replicator: Replicates combined trace from Host Funnel, to SoC Debug block or Host ETR
- Host APB interface: Allows access from the Host System memory map
- Host CPU APB interface (HOSTCPUDBG): Access to external debug registers of the Host CPU
- Host CPU Trace interface (HOSTCPUTRACE): Trace interface for Host CPU
- Host CTI Channel interface (HOSTCPUCTICHIN/HOSTCPUCTICHOUT): Connection with CTIs within the Host CPU.
- Two DAZ:
  - HOSTAUTH: Controls the level of debug allowed in the Host System, for either self-host or external debug agents. This DAZ is exposed on the HOSTDBGAUTH interface and is connected to all debug logic in the Host System.
  - HOSTEXTAUTH: Controls the level of debug access by an external debug agent. Used only for the Host APB AP.

The Host CPU debug logic is not part of SSE-700. SSE-700 provides an interface to integrate the debug logic, provided by the Host CPU cluster. SSE-700 mandates that the ETM is enabled in each Host CPU core.

The Host System debug provides expansion interfaces to allow for expansions and to integrate Host CPU. The expansion interfaces are:

- Host CPU expansion (Host CPU in the figure above):
  - APB, with 16MB of allocated memory space (HOSTCPUDBG)
  - CTI Channel interface (HOSTCPUCTICHIN/HOSTCPUCTICHOUT)
  - ATB interface (HOSTCPUTRACE). If there is more than a single CPU Core in the Host CPU cluster, an ATB funnel must be added.
- Host System expansion (Host Expansion in the figure above):
  - APB, with 16MB of allocated memory space (HOSTDBGAPBEXP)
  - CTI Channel interface (HOSTCTICHINEXP/HOSTCTICHOUTEXP)
  - ATB interface (HOSTDBGTRACEEXP)

Host System Debug on page 11-164 describes the memory map of the Host debug block.

7.3.5 External System {0-1} debug

The following figure shows the External System Debug, implemented by the External System Harness.

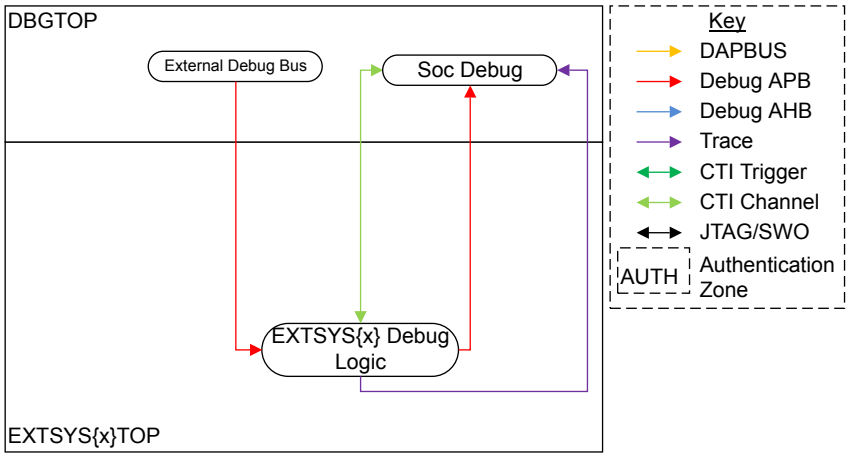

Figure 7-5 External System debug

The External System debug block is part of the External System Harness. The External System debug block includes the following:

- External System Debug APB (EXTSYS{0-1}DBG) interface: Provides the ability for the External System to integrate its own debug logic
- External System External Debug Access (EXTSYS{0-1}EXTDBG): Provides the ability for the External System {0-1} to be an external debug agent for another system in the SoC
- CTI Channel interface (EXTSYS{0-1}CTICHIN/EXTSYS{0-1}CTICHOUT): Expansion of the SSE-700 CTI network into the External System {0-1}
- Trace Interface: Expansion of the SSE-700 trace network into the External System {0-1}

The debug logic that is provided in the External System {0-1} (EXTSYS{x} Debug Logic) is IMPLEMENTATION DEFINED. No debug logic is also a valid configuration.

7.3.6 SoC debug

SSE-700 has some debug logic in the SoC Debug block, which is shared among all systems.

The following figure shows the SoC debug block:

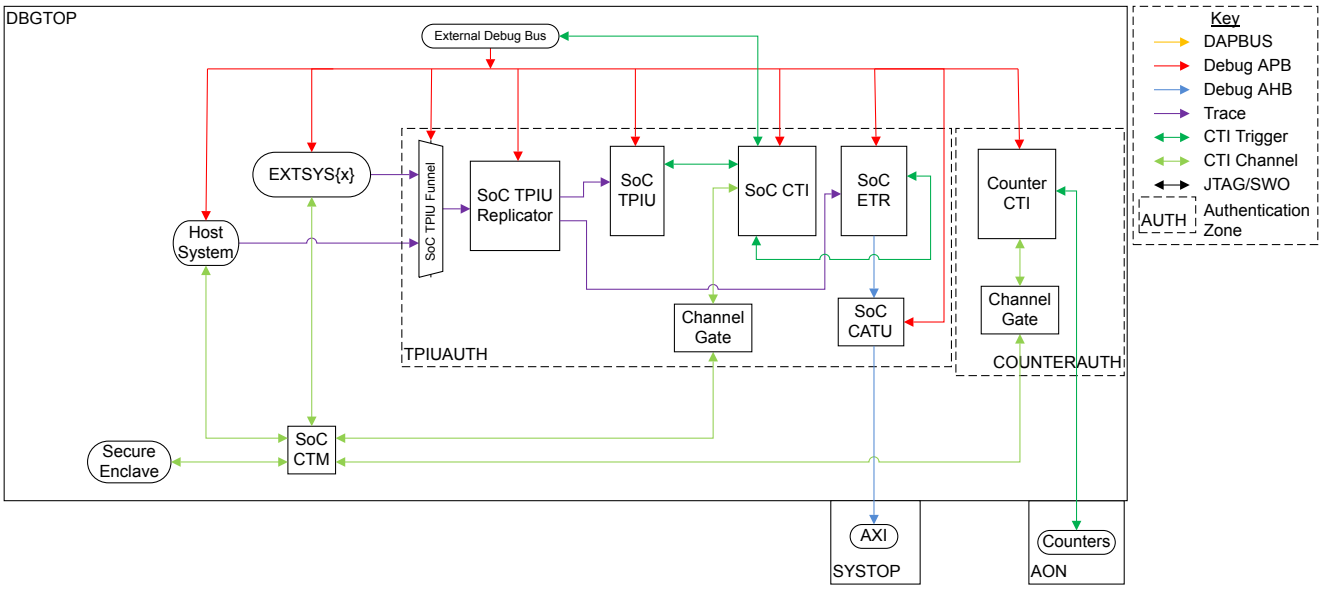

Figure 7-6 SoC debug logic

The SoC debug block contains the following:

- SoC TPIU: Provides trace off-chip
- SoC TPIU Funnel: Combines trace from all External Systems and the Host System
- SoC TPIU Replicator: Replicates the trace from the SoC TPIU Funnel to be sent
- SoC CTI: Provides triggers to and from the SoC TPIU, ETR, and DP
- SoC ETR: Allows trace to be routed to a memory location in the Host System
- SoC CATU: Allows for address translation of the SoC ETR AXI transactions
- SoC CTM: Combines CTI Channel interfaces from all External Systems and Host Systems
- Counter and TPIU Channel Gates
- Counter CTI: Provides triggers to and from the REFCLK and S32K counters.
- Two DAZ:
  - TPIUAUTH: Controls the level of debug that is provided by all logic in the SoC Debug, except for the Counter CTI
  - COUNTERAUTH: Controls the level of debug of the Counter CTI

11.1.2 External Debug Bus memory map on page 11-169 describes the memory map of the SoC Debug block.

7.4 Cross Trigger Infrastructure (CTI)

The following figure shows the interconnect between CTIs and CTMs.

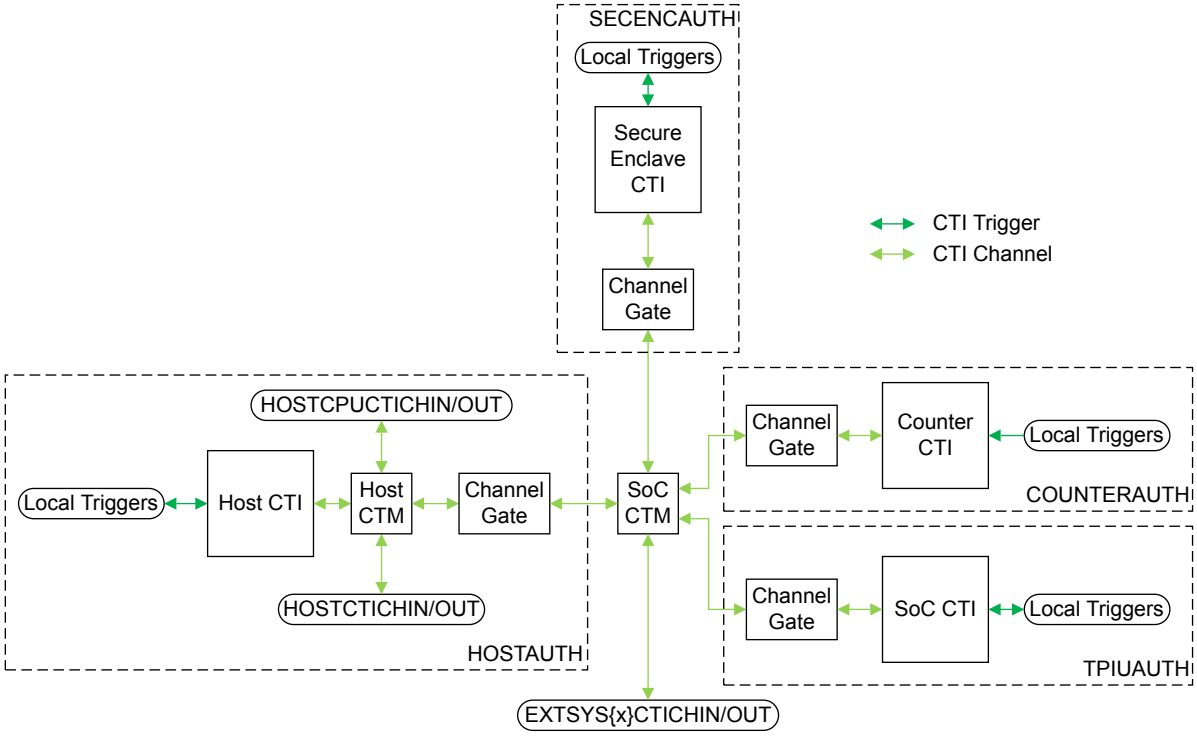

Figure 7-7 SSE-700 Cross Trigger Infrastructure

In SSE-700, there are the following CTIs and CTMs:

- Host CTI: Connected to Host ETR and Host STM
- Host CTM: Connects the Host CTI and SoC CTI. Also provides the following interfaces:
  - HOSTCPUCTICHIN and HOSTCPUCTICHOUT interfaces for the Host CPU
  - HOSTCTICHINEXP and HOSTCTICHOUTEXP interfaces for expansion of the Host debug logic
- SoC CTI: Connected to SoC TPIU, SoC ETR, and DP
- Counter CTI: Connected to the REFCLK and S32K Counter halt interfaces
- SoC CTM: Connects the SoC, Counter, Host, Secure Enclave CTIs and also provides the EXTSYS{0-1}CTICHIN and EXTSYS{0-1}CTICHOUT interfaces

Note

When a system exposes an event, it is globally broadcast to all systems. The debug agent is responsible for only exposing events which are safe to be exposed with all other systems.

SSE-700 also provides Channel Gates between the SoC CTM and the following components:

- Host CTM
- Secure Enclave CTI
- Counter CTI
- SoC CTI

The Channel Gate is placed on the Channel interfaces between CTI components to control whether events can be sent between CTI components. Each Channel Gates is controlled by a Channel Enable (CHEN) signal:

- HIGH: the Channel Gate is enabled and the events can be forwarded.
- LOW: the Channel Gate is disabled and the events can not be forwarded.

The CHEN is controlled by SCB from a Crypto Accelerator. For more information on SCB control, see the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

This section contains the following subsections:

7.4.1 Cross Trigger Interface (CTI)

The following tables show the assignment of trigger inputs and outputs for each CTI.

Table 7-2 Host CTI Trigger In

CTI Trigger In offset	Source component	Event type	Notes
0	Host STM TRIGOUTSPTE	Pulse	For more information on STM triggers, see the *Arm® CoreSight™ STM-500 System Trace Macrocell Technical Reference Manual*.
1	Host STM TRIGOUTSW		
2	Host STM TRIGOUTHETE		
3	Host STM ASYNCOUT		
4	Host ETR FULL	Level	For more information ETR triggers, see the *Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual*.
5	Host ETR ACQCOMP		
6	Host ETR FLUSHCOMP	Pulse	

Table 7-3 Host CTI Trigger Out

CTI Trigger Out offset	Sink component	Software handshake	Notes
0	Host STM HWEVENTS[0]	No	For more information on STM triggers, see the *Arm® CoreSight™ STM-500 System Trace Macrocell Technical Reference Manual*.
1	Host STM HWEVENTS[2]		
2	Host ETR TRIGIN		For more information ETR triggers, see the *Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual*.
3	Host ETR FLUSHIN		
4	Host GIC Interrupt 72		For more information on the connection to the Host GIC, see *11.2.1 Host CPU interrupt map* on page 11-172.
5	Host GIC Interrupt 73		

Table 7-4 SoC CTI Trigger In

CTI Trigger In offset	Source component	Event type	Notes
0	SoC TPIU FLUSHCOMP	Pulse	For more information on TPIU triggers, see the *Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual*.
1	SoC ETR FULL	Level	For more information ETR triggers, see the *Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual*.
2	SoC ETR ACQCOMP		
3	SoC ETR FLUSHCOMP	Pulse	

Table 7-5 SoC CTI Trigger Out

CTI Trigger Out offset	Sink component	Software handshake	Notes
0	TPIU TRIGIN	No	For more information on TPIU triggers, see the *Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual*.
1	TPIU FLUSHIN		
2	DP Event Status	Yes	For more information on DP triggers, see the *Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual*.
3	SoC ETR TRIGIN	No	For more information ETR triggers, see the *Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual*.
4	SoC ETR FLUSHIN		

Table 7-6 Counter CTI Trigger Out

CTI Trigger Out offset	Source component	Software handshake	Notes
0	**REFCLK** Counter Halt	No	-
1	**REFCLK** Counter Restart		
2	32K Counter Halt		
3	32K Counter Restart		

Table 7-7 Secure Enclave CTI Trigger In

CTI Trigger In offset	Source component	Event type	Notes
0	Secure Enclave Cortex-M0+ Halted	Pulse	-

Table 7-8 Secure Enclave CTI Trigger Out

CTI Trigger Out offset	Source component	Software handshake	Notes
0	Secure Enclave Cortex-M0+ External Debug Request	Yes	-
1	Reserved	-	
2	Secure Enclave Cortex-M0+ NVIC Interrupt 11	No	
3	Secure Enclave Cortex-M0+ NVIC Interrupt 12		
4	Reserved	-	
5			
6			
7	Secure Enclave Cortex-M0+ Debug Restart	No	

7.4.2 Cross Trigger Matrix (CTM)

SSE-700 provides a SoC and Host CTM, which connects all CTIs together.

The Host CTM connects the Host CTI to the SoC CTM and provides the following expansion interfaces:

- HOSTCTICHINEXP and HOSTCTICHOUTEXP: For expansion of the Host System debug
- HOSTCPUCTICHIN and HOSTCPUCTICHOUT: For connecting to the CTIs within the Host CPU

The SoC CTM connects the Host CTM, SoC CTI, and Counter Secure Enclave CTI together and provides the following expansion interface:

- EXTSYS{0-1}CTICHIN and EXTSYS{0-1}CTICHOUT: For connecting any CTIs and CTMs provided by the External System

7.4.3 CTI expansion

SSE-700 provides the following interfaces for expansion of the CTI:

- HOSTCTICHINEXP/HOSTCTICHOUTEXP for expansion of CTI within the Host System
- EXTSYS{0-1}CTICHIN/EXTSYS{0-1}CTICHOUT for expansion of CTI within the External System. Arm strongly recommends that if an External System implements CTI, it must provide a Channel Gate controlled by one of the expansion bits of the SCBs.
- HOSTCPUCTICHIN/HOSTCPUCTICHOUT for connection to a CTM within the Host CPU. Arm strongly recommends that the integrator of the SSE-700 uses these interfaces to integrate the CTI provided by the Host CPU.

7.5 Trace

SSE-700 provides a trace infrastructure to collect trace data from sources and transport it to the trace sinks.

SSE-700 provides the following trace components and interfaces :

- Sources:
  - *System Trace Macrocell* (STM)
  - Host CPU trace: Connected to the *Embedded Trace Macrocells* (ETMs) of the Host CPU
  - Host Debug block trace expansion: To be used to add other trace sources to the Host System
  - External System {0-1} trace expansion: To be used to add trace sources to the External System, for example, Cortex-M core ETM or ITM
- Funnels:
  - Host Funnel
  - SoC TPIU Funnel
- Replicators:
  - Host Replicator
  - SoC TPIU Replicator
- Sinks:
  - Host Embedded Trace Router
  - SoC Embedded Trace Router
  - SoC TPIU

SSE-700 requires an ETM for each Host CPU core implemented.

The following table shows the connection of the trace component input and output ports.

Table 7-9 Trace connectivity

Component	Input port	Source	Output port	Sink
STM	N/A	N/A	0	Host Funnel ATB port 0
Host CPU Funnel	0	Core 0 ETM	0	Host Funnel ATB port 1
	1	Core 1 ETM		
	2	Core 2 ETM		
	3	Core 3 ETM		
Host Funnel	0	System Trace Macrocell	0	Host Replicator
	1	Host CPU Funnel		
	2	HOSTDBGTRACEEXP		
Host Replicator	0	Host Funnel	0	SoC TPIU Funnel ATB port 0
			1	Host ETR
Host ETR	0	Host Replicator port 1	N/A	
SoC TPIU Funnel	0	Host Replicator	0	SoC TPIU Replicator
	1	EXTSYS0TRACEEXP		
	2	EXTSYS1TRACEEXP		
SoC TPIU Replicator	0	SoC TPIU Funnel	0	SoC TPIU
			1	SoC ETR

Table 7-9 Trace connectivity (continued)

Component	Input port	Source	Output port	Sink
SoC TPIU	0	SoC TPIU Replicator port 0	N/A	-
SoC ETR	0	SoC TPIU Replicator port 1		

7.6 System Trace Macrocell

The *System Trace Macrocell* (STM) is a high-bandwidth trace source for software instrumentation and enables hardware events to generate trace data.

The STM has an Extended Stimulus interface, occupying 16MB in the Host System memory map, and a configuration interface occupying 4KB in the Host System memory map. For more information on the Host System memory map, see *11.1.1 Host System memory map* on page 11-163. For more information on the System Trace Macrocell, see the *Arm® CoreSight™ STM-500 System Trace Macrocell Technical Reference Manual*.

Depending on the security of the master that generated the access to the Extended Stimulus interface, the STM generates trace packets with a different STPv2 Master ID. The following table shows the mapping of different masters in the SSE-700 to STPv2 Master ID.

Table 7-10 SSE-700 Master ID to STM STPv2 Master ID mapping

Master ID	Logical system master	STPv2 Master ID for secure accesses	STPv2 Master ID for non-secure accesses
0	Secure Enclave	0	64
1	Host CPU	1	65
4	AXI AP	4	68
32-63	IMPLEMENTATION DEFINED	32-63	96-127
Others	Default master	3	67

Note

Any STPv2 Master ID values that are not listed in the above table are Reserved.

The MasterID used by the STM is taken from the StreamID used by the Firewalls.

Access to the Extended Stimulus port of the STM when the DBGTOP power domain is in the OFF or WARM_RST power mode is treated as RAZ/WI.

Using its hardware event interface the STM can also be used to generate trace packets, based on hardware events. The following table shows the STM hardware events, sensitivity, and source.

Table 7-11 STM hardware events

STM event input	Edge/level	Source
0	Edge	Rising edge of Host CTI Trigger Out 0
1	Edge	Falling edge of Host CTI Trigger Out 0
2	Edge	Rising edge of Host CTI Trigger Out 1
3	Edge	Falling edge of Host CTI Trigger Out 1
4-31	Edge	Reserved
32-64	Level	Reserved

The **NSGUAREN** input of the STM is tied HIGH. This means that Secure and Non-secure accesses to the extended stimulus port behave the same.

7.7 ROM tables

SSE-700 has a CoreSight ROM table structure.

This structure is compliant with ADIv6, as the following figure shows:

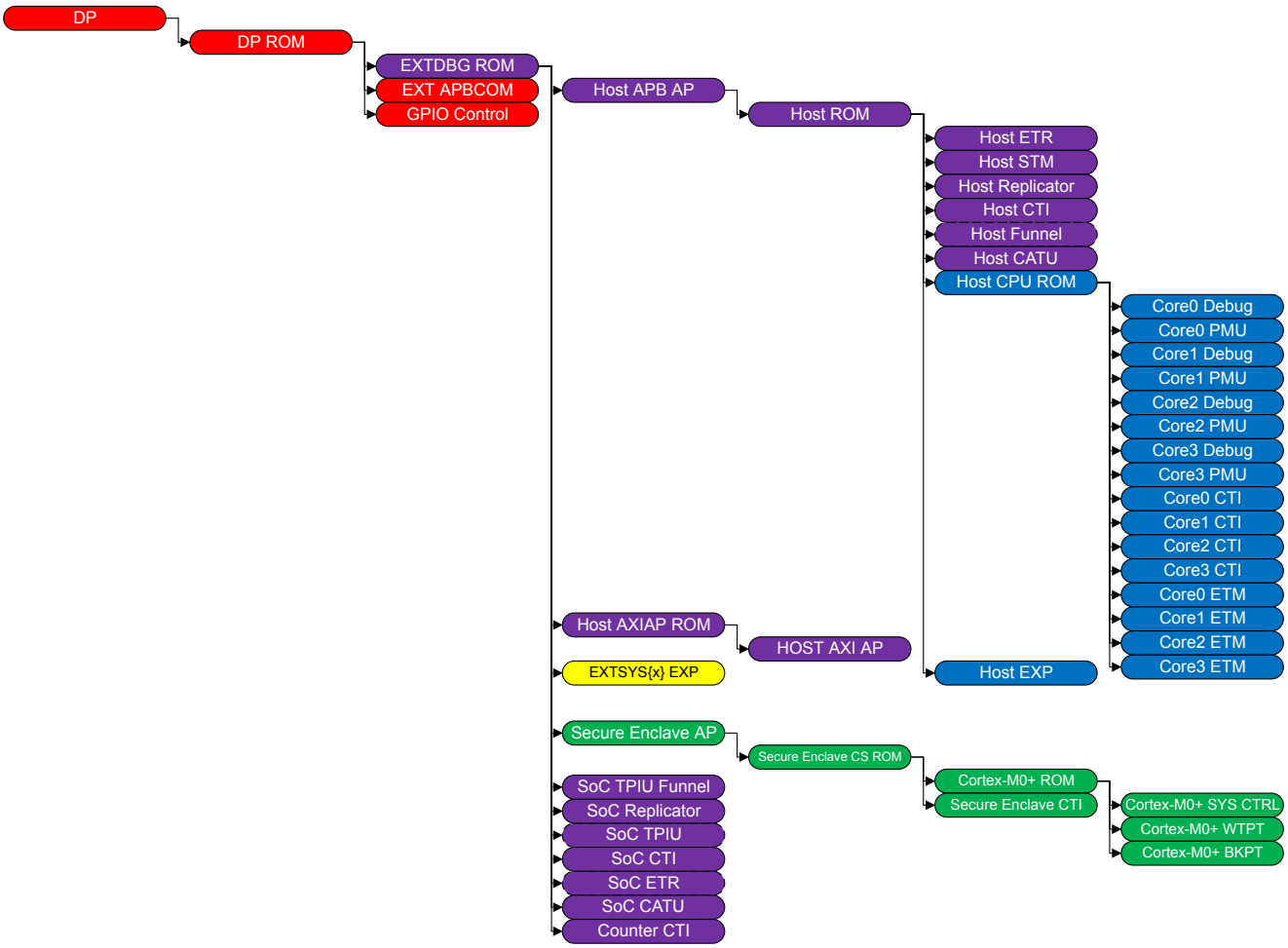

Figure 7-8 ROM table structure

The colors represent the power domain that the component resides in:

- Red: AONTOP
- Purple: DBGTOP
- Yellow: EXTSYS{0-1}TOP
- Blue: CLUSTOP
- Green: SECENCTOP

Note

For any components inside the SSE-700 which support both an internal and external view, all ROM table entries use the address of the external view.

SSE-700 conforms to the ADIv6 and includes a mix of Class 1 and 9 ROM tables. The following ROM tables, implemented by the SSE-700, are Class 9 ROM tables:

- DP ROM
- EXTDBG ROM

- Host ROM
- Host AXIAP ROM

All other ROM tables are implemented as Class 1.

SSE-700 also requires that the integrator adds additional ROM tables for the:

- External System, if it includes more than one debug component in the External System {x} Debug region of the External Debug Bus memory map
- Host CPU: SSE-700 supports the Cortex-A32, which already includes a ROM table.
- Host EXP ROM, if additional debug logic is added to the Host System

The additional ROM tables can be either Class 1 or 9 ROM tables, depending on the implementation. Arm strongly recommends that Class 9 ROM tables are used.

7.8 Granular Power Requestor (GPR)

SSE-700, within the system, uses the *Granular Power Requestor* (GPR) functionality of the ROM tables and the DP CTRL/STATUS register.

Some ROM tables provided by SSE-700 are Class 9 ROM tables with GPR functionality. Not all power and reset request signals from SSE-700 ROM tables and DP are used.

The following table shows what each signal from the component is used for:

Table 7-12 ROM table GPR connectivity

Component name	Signal name	Function
DP	**CDBGPWRUPREQ**	Request for **REFCLK** to be enabled
	CDBGPWRUPACK	Driven HIGH when REFCLK Q-Channel is in the Q_RUN state
	CDBGRSTREQ	Request reset of entire SoC. Signal is connected to Reset Controller.
	CDBGRSTACK	Acknowledge when SoC has been reset. Driven by the Reset Controller.
	CSYSPWRUPREQ	Reserved. Connected to **CSYSPWRUPACK** of the DP.
	CSYSPWRUPACK	Reserved. Connected to **CSYSPWRUPREQ** of the DP.
DP ROM	**CDBGPWRUPREQ0**	Power request for DBGTOP
	CDBGPWRUPACK0	Driven HIGH when DBGTOP is in the ON power mode
	CDBGPWRUPREQ[31:1]	Reserved
	CDBGPWRUPACK[31:1]	
	CSYSPWRUPREQ[31:0]	
	CSYSPWRUPACK[31:0]	
	CDBGRSTREQ	Request reset of DBGTOP
	CDBGRSTACK	Driven HIGH when DBGTOP is in WARM_RESET power mode
	CSYSRSTREQ	Request **nSRST**. Connected to Reset Controller.
	CSYSRSTACK	Acknowledge when SoC has been reset and is held before Secure Enclave starts to boot. Driven by Reset Controller.
SDC-600	**REMPUR**	Power request for SYSTOP
	REMPUA	Driven HIGH when SYSTOP is in the ON or FUNC_RET power mode
	REMRR	Connected to SDC-600 **REMRA**
	REMRA	Connected to SDC-600 **REMRR**

Table 7-12 ROM table GPR connectivity (continued)

Component name	Signal name	Function
EXTDBG ROM	**CDBGPWRUPREQ0**	Power request for SECENCTOP
	CDBGPWRUPACK0	Driven HIGH when the SECENCTOP is in ON
	CDBGPWRUPREQ[2:1]	Power request for External System {0-1}. This signal is part of the External System Harness, EXTSYS{0-1}PWRREQ interface. When the associated External System is not implemented, then the signal is Reserved – Connected to the respective **CDBGPWRUPACK**.
	CDBGPWRUPACK[2:1]	Acknowledge when the External System has entered a power mode where accesses to the debug components will be allowed. This signal is part of the External System Harness EXTSYS{0-1}PWRREQ interface. When the associated External System is not implemented, then the signal is Reserved: Connected to respective **CDBGPWRUPREQ**.
	CDBGPWRUPREQ[31:3]	Reserved
	CDBGPWRUPACK[31:3]	
	CSYSPWRUPREQ[31:0]	
	CSYSPWRUPACK[31:0]	
	CDBGRSTREQ	Reserved. Connected to **CDBGRSTACK**.
	CDBGRSTACK	Reserved. Connected to **CDBGRSTREQ**.
	CSYSRSTREQ	Reserved. Connected to **CSYSRSTACK**.
	CSYSRSTACK	Reserved. Connected to **CSYSRSTREQ**.
Host ROM	**CDBGPWRUPREQ0**	Power request for Host CPU power domain. This signal is connected to the CLUSTOP PPU.
	CDBGPWRUPACK0	Acknowledge when the Host CPU is accessible. This signal is driven by the CLUSTOP PPU, when the domain is in the ON or FUNC_RET power modes.
	CDBGPWRUPREQ[31:1]	Reserved
	CDBGPWRUPACK[31:1]	
	CSYSPWRUPREQ[31:0]	
	CSYSPWRUPACK[31:0]	
	CDBGRSTREQ	Reserved. Connected to **CDBGRSTACK**.
	CDBGRSTACK	Reserved. Connected to **CDBGRSTREQ**.
	CSYSRSTREQ	Reserved. Connected to **CSYSRSTACK**.
	CSYSRSTACK	Reserved. Connected to **CSYSRSTREQ**.

Table 7-12 ROM table GPR connectivity (continued)

Component name	Signal name	Function
Host AXIAP ROM	**CDBGPWRUPREQ[31:0]**	Reserved
	CDBGPWRUPACK[31:0]	
	CSYSPWRUPREQ0	Power request for SYSTOP to be in ON or FUNC_RET. This signal is connected to the SYSTOP PPU.
	CSYSPWRUPACK0	Acknowledge when SYSTOP is in the ON or FUNC_RET power mode
	CSYSPWRUPREQ1	Power request for CLUSTOP to be in ON or FUNC_RET. This signal is connected to the CLUSTOP PPU.
	CSYSPWRUPACK1	Acknowledge when CLUSTOP is in the ON or FUNC_RET power mode
	CSYSPWRUPREQ[3:2]	Power request for EXTSYS{0-1}TOP power domain. When the associated External System is not implemented, then the signal is Reserved, connected to respective **CDBGPWRUPACK**.
	CSYSPWRUPACK[3:2]	Acknowledge for EXTSYS{0-1}TOP power domain is in a power mode to be able to process transactions. When the associated External System is not implemented, then the signal is Reserved, connected to respective **CDBGPWRUPREQ**.
	CSYSPWRUPREQ[15:4]	Reserved
	CSYSPWRUPACK[15:4]	
	CSYSPWRUPREQ[31:16]	IMPLEMENTATION DEFINED. Exposed in the HOSTDBGPWRREQ interface. The integrator uses these signals for any other power domains, which can be accessed by the Host AXI AP.
	CSYSPWRUPACK[31:16]	IMPLEMENTATION DEFINED. Exposed in the HOSTDBGPWRREQ interface. The integrator uses these signals to indicate when the power domain powered.
	CDBGRSTREQ	Reserved. Connected to **CDBGRSTACK**.
	CDBGRSTACK	Reserved. Connected to **CDBGRSTREQ**.
	CSYSRSTREQ	Reserved. Connected to **CSYSRSTACK**.
	CSYSRSTACK	Reserved. Connected to **CSYSRSTREQ**.

For the **CSYSPWRUPREQ/ACK** and **CDBGPWRUPREQ/ACK** interfaces of the ROM tables which are described as Reserved in the table above, the **REQ** is looped back to the **ACK**. The value in the ROM table DEVID.NUMREQ field reflects the number of **REQ/ACK** signals implemented by the ROM table.

7.9 CoreSight timestamp

SSE-700 supports a CoreSight timestamp. However, its control and operation frequency is IMPLEMENTATION DEFINED.

7.10 GPIO control

The GPIO Control is a peripheral on the External Debug bus which is only accessible through the DP. Access from any other masters treats the location as Reserved.

The GPIO Control enables a debugger to control the CALC interface. See *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

Chapter 8
Interconnect

This chapter describes the Host System Interconnect, which connects masters and slaves of the Host System.

It contains the following sections:

8.1 NIC

SSE-700 contains an internal fabric responsible for connecting all masters and slaves within SSE-700. This internal fabric is created using CoreLink NIC-400 switches.

SSE-700 contains the following switches:

- NIC (Main)
- NIC (AONPERIPH)
- NIC (SYSPERIPH)
- NIC (DBGPERIPH)

The following figure shows the connection between the NIC switches.

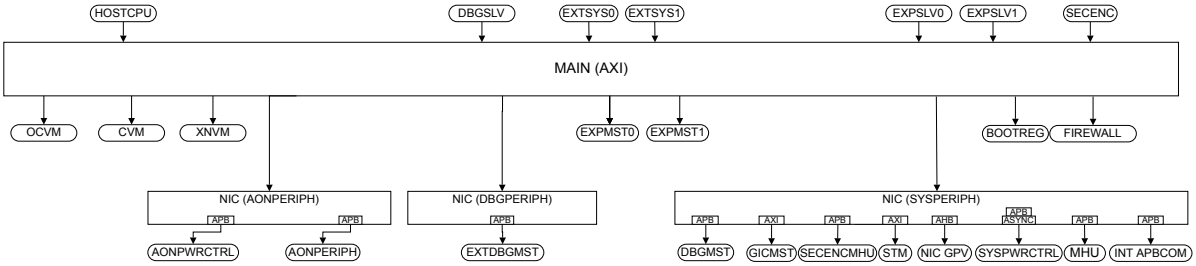

Figure 8-1 Interconnect interfaces

The Main NIC is the main switch connecting all the masters, both internal and external to SSE-700, to the slaves of the SoC. For some slaves, access is via one of the peripheral NICs (AONPERIP / SYSPERIPH / DBGPERIPH).

The peripheral NICs (AONPERIPH/SYSPERIPH/DBGPERIPH) provide access to peripherals in the AONTOP, SYSTOP, and DBGTOP power domains. Some of the ports on these NICs provide protocol conversion to APB and implement clock domain crossing.

NIC(Main) includes a *Global Programmers View* (GPV).

The NIC interconnect is not a full crossbar and certain masters can only access certain slaves as the following table shows:

Table 8-1 Slave access to master interfaces of interconnect

Master interface	Address ranges	Slave interface					
		HOST CPU	GIC SLV	DBG SLV	EXT SYS {0-1}	SEC ENC	EXP SLV {0-1}
BOOTREG	0x0000_0000 – 0x0000_0FFF	Y	N	Y	N	Y	N
CVM	0x0200_0000 – 0x03FF_FFFF		Y	Y	Y	Y	Y
XNVM	0x0800_0000 – 0x0FFF_FFFF		Y	Y	Y	Y	Y
DBGMST	0x1000_0000 – 0x17FF_FFFF		N	N	N	Y	N
EXTDBGMST	0x1800_0000 – 0x19FF_FFFF		N	N	N	Y	N
AONPWRCTRL	0x1A02_0000 – 0x1A04_FFFF		N	Y	N	Y	N
AONPERIPH	0x1A00_0000 – 0x1A01_FFFF 0x1A20_0000 – 0x1A6F_FFFF		N	Y	Y	Y	Y
FIREWALL	0x1A80_0000 – 0x1A9F_FFFF		N	Y	N	Y	N
MHU	0x1B00_0000 – 0x1B0F_FFFF		N	Y	N	Y	N
SECENCMHU	0x1B80_0000 – 0x1B83_FFFF		N	Y	Y	Y	N
INT APBCOM	0x1B90_0000 – 0x1B90_FFFF		N	Y	N	Y	N
SYSPWRCTRL	0x1BC0_0000 – 0x1BC4_FFFF		N	Y	N	Y	N
GICMST	0x1C00_0000 – 0x1CFF_FFFF		N	Y	Y	Y	Y
STM	0x1D00_0000 – 0x1DFF_FFFF		N	Y	N	Y	Y
NIC GPV	0x1E00_0000 – 0x1E0F_FFFF		N	Y	N	Y	N
EXPMST{0-1}	0x4000_0000 – 0x7FFF_FFFF		Y	Y	Y	Y	Y
OCVM	0x8000_0000 – 0xFFFF_FFFF		Y	Y	Y	Y	Y

This section contains the following subsections:

- *8.1.1 Attributes of the AXI master interface* on page 8-126.
- *8.1.2 Attributes of the AXI slave interface* on page 8-127.

8.1.1 Attributes of the AXI master interface

The following table lists AXI ID width and read/write issuing capability of the SSE-700 master AXI interfaces.

Table 8-2 AXI master interface attributes

Interface	Read issuing capability	Write issuing capability	AXI ID width
CVM	32	32	12
XNVM			
OCVM			
EXPMST{0-1}			

8.1.2 Attributes of the AXI slave interface

The following table lists the AXI ID width and the read/write issuing capability of the SSE-700 slave AXI interfaces.

Table 8-3 AXI slave interface attributes

Interface	Read issuing capability	Write issuing capability	AXI ID width
HOST CPU	32	16	8
EXTSYS 0	8	8	8
EXTSYS 1			
SECENC	4	4	4
EXPSLV{0-1}	32	16	8

8.2 Quality of Service (QoS)

The SSE-700 Main NIC-400 implementation supports a programmable QoS scheme for the Host CPU, External Systems {0-1}, Debug AXI, and Secure Enclave interfaces.

Also, the Expansion Slaves {0-1} provide QoS values from the protocol interface ("From Master"):

- "Programmable" for Host CPU, External Systems {0,1}, Debug AXI and Secure Enclave:
  - The interface does not include the **ARQOS** and **AWQOS** signals.
  - The QoS setting is configured via the GPV.
- "From master" for the Expansion Slave {0,1}: In this case, the interface has the **ARQOS** and **AWQOS** signals.

The following interfaces have the **ARQOS** and **AWQOS** signals included:

- CVM
- XNVM
- OCVM
- HOSTEXPMST{0-1}

No QoS functionality is supported on any of the other NICs in SSE-700.

To support the programmable option for QoS, the *Global Programmers View*(GPV) has been enabled for the respective slave interfaces. The address offsets from the NIC-400 Main GPV base are listed below:

- `0x42000-0x42FFF`: HOSTCPU
- `0x43000-0x43FFF`: EXTSYS0
- `0x44000-0x44FFF`: EXTSYS1
- `0x45000-0x45FFF`: SECENC
- `0x46000-0x46FFF`: DBGSLV

8.3 Firewall

The Firewall has been implemented within the subsystem to provide both monitoring and protection of the address space, between the many different entities within the SSE-700.

This section describes the SSE-700 Host System Firewall configurations and the IMPLEMENTATION DEFINED behavior of the Firewall. For the full set of generic Firewall features, see *Appendix C Firewall* on page Appx-C-281.

This section contains the following subsections:

8.3.1 Firewall Component interfaces

The Firewall Component includes clock, reset, bypass, DFT control, AXI4 Stream configuration, AXI5 Master and Slave, and Q-Channel power and clock control interfaces.

The following figure shows the *Firewall Component* (FC) interfaces.

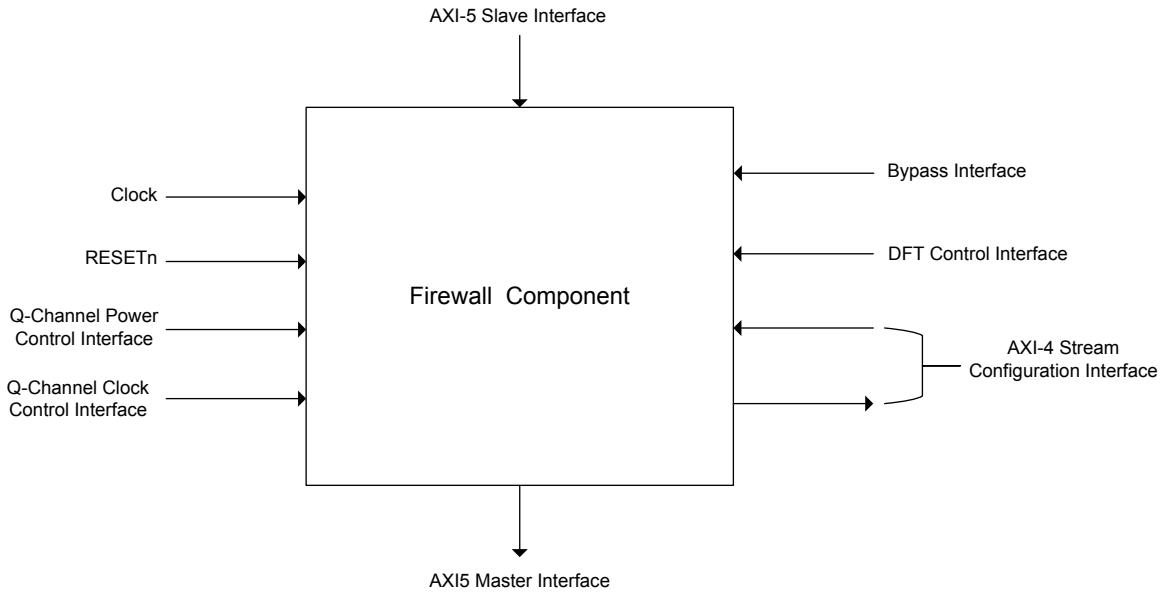

Figure 8-2 FC interfaces

AXI5 Slave and Master interfaces

The slave interface is the ingress port where a transaction enters the FC, and the master interface is the egress where a transaction exits.

The interfaces have the following properties:

- AMBA AXI5 protocol with the following properties set to true:

— Wakeup_Signal
— Untranslated_Transaction

AXI4 Stream Configuration interface

This configuration interface is bidirectional, and is used for communication between the *Firewall Controller* (FCTLR) and the FCs.

The interface is implemented based on the *AMBA® 4 AXI4-Stream Protocol Specification*, in which messages can be initiated from either side.

Low Power interfaces

The Firewall Component has two low power Q-Channel interfaces.

The low power interfaces are:
- A Q-channel for clock control
- A Q-channel for power gating.

The low power states implemented in the Firewall are based on the states defined in the *Arm® Power Control System Architecture Specification*.

Bypass interface

The Bypass interface serves to determine the behavior of protection logic checks that are applied by the Firewall on transactions.

In the SSE-700, the Bypass interface of the Host System Firewall is driven by an SCB bit.

For more information on the Bypass interface, see *Appendix C Firewall* on page Appx-C-281.

8.3.2 Firewall Controller interfaces

The Firewall Controller includes clock, reset, interrupt, bypass, lockdown, protection, DFT control, AXI4 Stream configuration, AXI5 programming, P-Channel Power Control, and Q-Channel Clock control interfaces.

The following figure shows the *Firewall Controller* (FCTLR) interfaces.

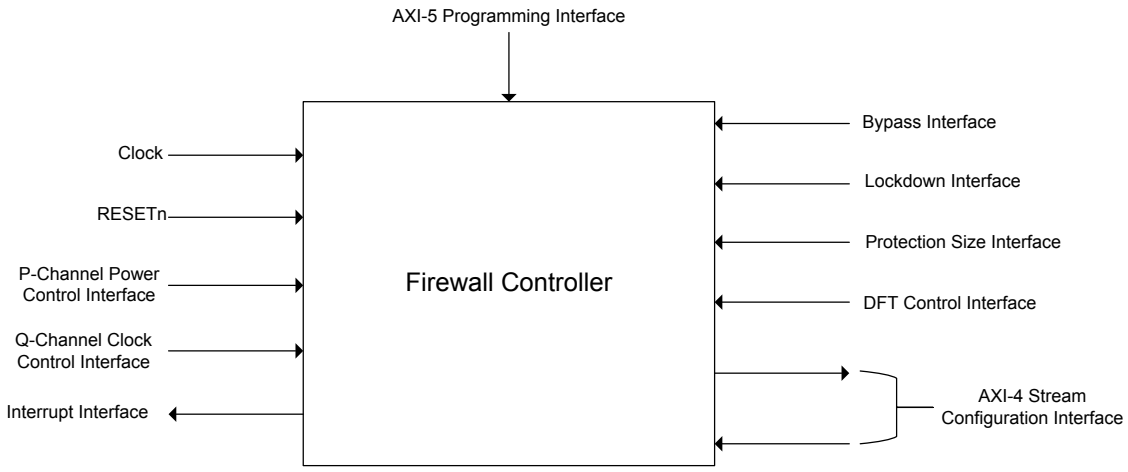

Figure 8-3 FCTLR interfaces

AXI5 Programming interface

This interface is used to program the Firewall configuration registers inside the FCTLR and FCs.

The interface has the following properties:

- AMBA AXI5 protocol with the following properties set to true:
  - Wakeup_Signal
  - Untranslated_Transaction

For more information on supported configuration access through this interface to the Firewall Controller, see *8.3.3 Implementation defined behavior* on page 8-131.

AXI4 Stream Configuration interface

This configuration interface is bidirectional, and is used for communication between the FCTLR and the FCs.

For more information, see *AXI4 Stream Configuration interface* on page 8-130.

Lockdown interface

The lockdown input controls the lockdown extension functionality of the Firewall.

For more information on the lockdown behavior, see *Appendix C Firewall* on page Appx-C-281.

Interrupt interface

The Firewall Controller implements two interrupt interfaces.

The two interrupt interfaces are for:

- Firewall interrupt: A single interrupt signal for the entire Firewall
- Tamper interrupt: A separate interrupt interface to generate tamper interrupt to the system

Protection Size interface

The Protection Size interface is an input vector which is organized as a single dimensional array in which each 8 bits is allocated to one FC.

In the SSE-700, the sampled values on the protection interface of the Host FCs are defined in *8.3.5 Host System Firewall* on page 8-134.

Low Power interfaces

The Firewall Controller has two low power interfaces.

These low power interfaces are:

- A Q-Channel interface for clock control
- A P-Channel interface for power gating. The P-Channel supports the following power modes: OFF, FUNC_RET, ON.

The low power states implemented in Firewall are based on the states defined in *Arm® Power Control System Architecture Specification*.

Bypass interface

The Bypass interface serves to determine the behavior of protection logic checks that are applied by the Firewall on transactions.

The FCTLR Bypass interface behaves in a same way as FC Bypass interface. For more information, see *Bypass interface* on page 8-130.

8.3.3 Implementation defined behavior

This section defines IMPLEMENTATION DEFINED behaviors that are visible to software or impact on the SSE-700.

Appendix C Firewall on page Appx-C-281 describes behaviors that are IMPLEMENTATION DEFINED. This section applies to both the Host System and Secure Enclave Firewalls, unless otherwise stated.

Note

This section must be read with *Appendix C Firewall* on page Appx-C-281.

- The Firewall always occupies 2MB with any unused 64KB pages, allocated to Firewall Components, being marked as Reserved.
- Bus Slave, Bus Master, and Programming interfaces:
  - Burst based with address that is reported in the Fault Entries and Error Detection Reports using the starting address of the transaction.
  - There is one Bus Slave and Master interface per Firewall Component which supports the same address, data, and transaction properties.
  - Any transaction that is received on the Bus Slave interface is forwarded to the Bus Master interface if it passes the protection logic checks or the Firewall Bypass interface is asserted.
- Transaction processing, for protection and monitoring:
  - Firewall Components process transactions for read and write transactions separately.
  - There are no guarantees about the order in which Fault Entries or Error Detection Reports are generated:
    - Between read and write transactions
    - Between read or write transactions which have different AXI IDs
- Firewall Controller only implements region 0 to 2 as required by the specification.
- Shadow Registers, when SRE.1 is implemented.
  - Implemented as SRAM
  - Support memory retention
  - Applies compression
- When a Firewall Component (including the Firewall Controller), which supports PE.1 or greater, terminates a transaction it:
  - Generates an AMBA AXI5 DECERR, if PE_ST.ERR is set to `0b1`.
  - Generate StreamID specific read data responses, if PE_ST.RAZ is set to `0b0`, as defined by the `FC_ERR_RESP_DEF` or `FIREWALL_F0_CFG_GLOBAL_SSE700_ERR_RESP_PER_MST_ID{x}_VAL` at design time, where x is the StreamID value.
  - Mark the response as being generated by the Firewall using the RUSER[0] or BUSER[0] bits.
- When a Firewall Component, which supports ME.2, detects an error transaction it:
  - Generate StreamID specific read data responses, if ME_ST.RDUM is set to `0b0`, as defined by the `FC_ERR_RESP_DEF` or `FIREWALL_F0_CFG_GLOBAL_SSE700_ERR_RESP_PER_MST_ID{x}_VAL` at design time, where x is the StreamID value.
  - Mark the response as being generated by the Firewall using the RUSER[0] or BUSER[0] bits.
- The Firewall Controller only supports configuration accesses with the following properties, otherwise a Configuration Access Error is generated:
  - 32-bit word-aligned access only.
  - Memory type must be either Device-nRnGnE or Device-nRnGE.
- The Firewall Controller does not support any exclusive accesses and treats the access as normal.
- The Firewall Controller responds to a configuration access, which generates a Configuration Access Error by:
  - Generating an AMBA AXI5 SLVERR, if FW_ST.ERR is set to `0b1`.
  - Generate StreamID specific read data responses, if FW_ST.RAZ is set to `0b0`, as defined by the `FC_ERR_RESP_DEF` or `FIREWALL_F0_CFG_GLOBAL_SSE700_ERR_RESP_PER_MST_ID{x}_VAL` at design time, where x is the StreamID value.
- Reset value of the PE_CTRL.FLT_CFG is `0b10`.
- Reset value of the PE_CTRL.RAZ is `0b0`.
- Reset value of the PE_CTRL.ERR is `0b1`.

All Firewall Components, in the SSE-700 uses the AMBA AXI5 bus protocol. The Firewall:

- Ignores the transient property.
- Treats any memory which is not Write-Back cacheable at both inner and outer as Non-cacheable.

The following table shows a summary of the mapping between the Firewall, RGN_TCFG2.MA field, and the AMBA AXI5 memory types. The table also shows how the Firewall handles the AxLOCK signal on the incoming transaction.

Note

The following syntax is used:

- I: Inner
- o: Outer
- NC: Non-cacheable
- WT: Write-Through cacheable
- WB: Write-Back cacheable

Table 8-4 Firewall memory attribute to AMBA AXI5 AxCACHE mapping

Firewall memory attribute	AMBA AXI5 AxCACHE	AxLOCK	Note
Device-nGnRnE	Device Non-bufferable	As incoming transaction	-
Device-nGnRE Device-nGRE Device-GRE	Device Bufferable		
Normal-iNC-o{NC,WT,WB} Normal-iWT-o{NC,WT,WB} Normal-iWB-o{NC,WT}	Normal Non-Cacheable Bufferable		
Normal-iWB-oWB	Write-Back No Allocate Write-Back Read-Allocate Write-Back Write-Allocate Write-Back Read and Write-Allocate	Set to 0	The value that is selected depends on the read and write allocation policy that is defined in the RGN_TCFG2.MA field for the inner domain.

The following table shows a summary of the mapping between the AMBA AXI5 memory types and the Firewall memory attribute types.

Table 8-5 AMBA AXI5 AxCACHE to Firewall memory attribute mapping

AMBA AXI5 AxCACHE	Firewall memory attribute	Notes
Device Non-bufferable	Device-nGnRnE	-
Device Bufferable	Device-nGnRE	
Normal Non-Cacheable Bufferable Normal Non-Cacheable Non-Bufferable Write-Through No Allocate Write-Through Read-Allocate Write-Through Write-Allocate Write-Through Read and Write-Allocate	Normal-iNC-oNC	
Write-Back No Allocate Write-Back Read-Allocate Write-Back Write-Allocate Write-Back Read and Write-Allocate	Normal-iWB-oWB	The read and write allocation policy, for inner and outer, is set to the same value as the AXI5 AxCACHE fields.

8.3.4 Firewall read response value

This section describes how to configure the Firewall to set the read data response to a value based on the StreamID.

The Firewall can be configured to set the read data response to a value based on the StreamID when one of the following has occurred:

- A transaction is terminated by the protection logic of the Firewall.
- A bus error is detected by the monitoring logic of the Firewall.
- A Configuration Access Error has been generated.

The read data response value that is returned is a design time configuration, set using the following parameters:

- `FC_ERR_RESP_DEF`
- `FIREWALL_F0_CFG_GLOBAL_SSE700_FC_MST_ID{x}_VAL`
- `FIREWALL_F0_CFG_GLOBAL_SSE700_ERR_RESP_PER_MST_ID{x}_VAL`

The Firewall determines which value is returned by performing the following:

- If SINGLE_MST is set to 1, then return FC_ERR_RESP_DEF.
- Looking up the MasterID from the StreamID of the transaction against the values of `FIREWALL_F0_CFG_GLOBAL_SSE700_FC_MST_ID{x}_VAL` configuration option.
  - If the value is found, then the corresponding value in `FIREWALL_F0_CFG_GLOBAL_SSE700_ERR_RESP_PER_MST_ID{x}_VAL` is returned.

    Note

    `FIREWALL_F0_CFG_GLOBAL_SSE700_FC_MST_ID{x}_VAL` and `FIREWALL_F0_CFG_GLOBAL_SSE700_ERR_RESP_PER_MST_ID{x}_VAL` are matched based on the value of x.
  - If the value is not found, then the value that is defined by FC_ERR_RESP_DEF is returned.

8.3.5 Host System Firewall

The Host System Firewall has the following configurations:

- Lockdown Extension level 2
- Save and Restore Extension level 1
- Security Extension level 1

Note

SSE-700 contains two Firewalls, in the Secure Enclave and the Host System. This section applies only to the Host System Firewall.

The Host System Firewall is distributed across multiple clock and power domains, with the Firewall Controller located in AONTOP.

All transactions issued on the slave interfaces of the interconnect, must pass through at least one Firewall Component of the Host System Firewall before being issued on the master interfaces of the interconnect. The location of the Firewall Components within SSE-700 is shown in the following figure.

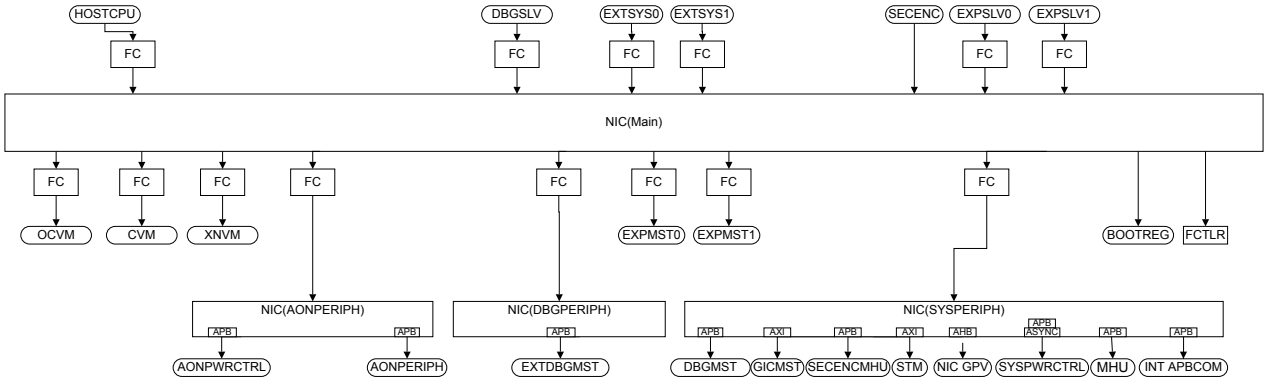

Figure 8-4 Location of Firewall Components

The following table shows the location of the Firewall Components and the mapping to the Firewall Component IDs. The following table shows location names.

Table 8-6 Host System Firewall Component locations

Firewall Component	Location	Firewall Component ID
Firewall Controller	Firewall	0
SYSPERIPH	Between NIC(Main) and NIC(SYSPERIPH)	1
DBGPERIPH	Between NIC(Main) and NIC(DBGPERIPH)	2
AONPERIPH	Between NIC(Main) and NIC(AONPERIPH)	3
XNVM	XNVM	4
CVM	CVM	5
HOSTCPU	HOSTCPU	6
EXTSYS0	EXTSYS0	7
EXTSYS1	EXTSYS1	8
EXPSLV0	EXPSLV0	9
EXPSLV1	EXPSLV1	10
EXPMST0	EXPMST0	11
EXPMST1	EXPMST1	12

Table 8-6 Host System Firewall Component locations (continued)

Firewall Component	Location	Firewall Component ID
OCVM	OCVM	13
Debug	DBGSLV	14

SSE-700 allows an implementation to configure certain aspects of the Firewall Components of the Host System Firewall.

The following table shows the configuration values of the Firewall Components. Where the value is fixed, an explicit value is provided. Where the value is configurable, a range of values is provided.

Table 8-7 Host System Firewall Component configuration

Firewall Component	PE_LV	ME_LVL	TE_LVL	RSE_LVL	NUM_RGN	MNRS	MXRS	NUM_MPE	SINGLE_MST
FCTLR	1	0	0	1	3	7	21	1	0
SYSPERIPH	1	0	0	0	22	7	29	1	0
DBGPERIPH	1	0	0	0	1	7	29	1	0
AONPERIPH	1	0	0	0	40	7	23	1	0
XNVM	2	0	0	0/1	16/32/48/64	7	27	4	0
CVM	2	0	0	0/1	16/32/48/64	3	25	4	0
HOSTCPU	0	2	0	0	NA	NA	NA	NA	1
DBG	2	2	2	1	4/8	7	32	2	0
EXTSYS{0-1}	2	2	2	1	8/16	7	32	1	1
EXPSLV{0-1}	2	2	2	1	8/16/32	7	32	4	0
EXPMST{0-1}*	2	0	0	0/1	8/16/32	3	29	4	0
	1	0	0	0	1-64	7	29	4	0
OCVM	2	0	0	0/1	16/32/64	7	31	4	0

* The allowed values depend on the level of Protection Extension implemented. EXMPST0 can be configured to support either PE.1 or PE.2. EXPMST1 only supports PE.2

The following configurable values always apply for all Firewall Components:

- MST_ID_WIDTH is 8
- SEC_SPT is `0b1`
- MA_SPT is `0b1`
- SH_SPT is `0b0`
- INST_SPT is `0b1`
- PRIV_SPT is `0b1`

1 fault entries and 1 error detection report are implemented by each Firewall .

For Firewall Components that implement PE.1, the following regions are implemented:

- Firewall Controller:
  - As defined in *Appendix C Firewall* on page Appx-C-281
  - Configuration Master is set to the StreamID of the Secure Enclave.
- SYSPERIPH:
  - 1 region for each Host External System MHU
  - 1 region for CLUSTOP PPU

  - 1 region for each CORE{0-3} PPU
  - 1 region for INT APBCOM
  - 1 region for each MHU in the Secure Enclave
  - 1 region for STM Extended Stimulus port
  - 1 region for NIC GPV
  - 1 region for the GIC
  - 1 region Host Debug address space
- DBGPERIPH:
  - 1 region External Host Debug address space
- AONPERIPH:
  - 1 region for FW RAM PPU
  - 1 region for SYSTOP PPU
  - 1 region for DBGTOP PPU
  - 1 region for Interrupt Router
  - 1 region for each S32K CNTBase{0-1}
  - 1 region for S32K CNTRead
  - 1 region for S32K CNTControl
  - 1 region for S32K CNTCTL
  - 1 region for each **REFCLK** CNTBase{0-3}
  - 1 region for **REFCLK** CNTRead
  - 1 region for **REFCLK** CNTControl
  - 1 region for **REFCLK** CNTCTL
  - 1 region for Non-secure WDOG Refresh
  - 1 region for Non-secure WDOG Control
  - 1 region for Secure WDOG Refresh
  - 1 region for Secure WDOG Control
  - 1 region for Host Base System Control
  - 1 region for System ID
  - 1 region for each UART{0-1}
  - 16 regions allocated to the AON Expansion. The size and base address of these regions is IMPLEMENTATION DEFINED, but must be allocated exclusively within the AON Expansion address space.
- EXPMST0:
  - IMPLEMENTATION DEFINED when the EXPMST0 Firewall Component is configured with a PE_LVL of 1.

The **SOCCFG** interface signals, drive the Protection Size interface of the following Firewall Components:

- **CVMSIZE** is connected to the CVM Firewall Component.
  - Legal values are: 256KB, 512KB, 1MB, 2MB, 4MB, 8MB, 16MB, 32MB.
  - Any values less than 256KB are treated as if 256KB was set.
  - Any values greater than 32MB are treated as if 32MB was set.
- **XNVMSIZE** is connected to the XNVM Firewall Component.
  - Legal values are: 4MB, 8MB, 16MB, 32MB, 64MB, 128MB.
  - Any values less than 4MB are treated as if 4MB was set.
  - Any values greater than 128MB are treated as if 128MB was set.
- **OCVMSIZE** is connected to the OCVM Firewall Component.
  - Legal values are: 0B, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB, 256MB, 512MB, 1GB, 2GB.
  - Any values less than 2MB, other than 0B, are treated as if 2MB was set.
  - Any values greater than 2GB are treated as if 2GB was set.

Note

See *Appendix C Firewall* on page Appx-C-281 for more information on the Protection Size interface.

For all other Firewall Components with the Protection Size interface, the interface is tied off internal to SSE-700 to match the address width of the Firewall Component.

8.3.6 Host System Firewall regions

AONPERIPH and SYSPERIPH Firewall Components are pre-defined regions.

AONPERIPH Firewall Component regions

The following table shows the AONPERIPH Firewall Component regions.

Table 8-8 AONPERIPH Firewall Component regions

Region number	Peripheral	Field	Value
0	System ID	Base Address	0x1A00_000
		Size	0x0C
		MULnPO2	0b0
1	Host Base System Control	Base Address	0x1A01_0000
		Size	0x0C
		MULnPO2	0b0
2	FW PPU	Base Address	0x1A02_0000
		Size	0x0C
		MULnPO2	0b0
3	SYSTOP PPU	Base Address	0x1A03_0000
		Size	0x0C
		MULnPO2	0b0
4	DBGTOP PPU	Base Address	0x1A04_0000
		Size	0x0C
		MULnPO2	0b0
5	REFCLK CNTControl	Base Address	0x1A20_0000
		Size	0x0C
		MULnPO2	0b0
6	REFCLK CNTRead	Base Address	0x1A21_0000
		Size	0x0C
		MULnPO2	0b0
7	REFCLK CNTCTL	Base Address	0x1A22_0000
		Size	0x0C
		MULnPO2	0b0
8	REFCLK CNTBase0	Base Address	0x1A23_0000
		Size	0x0C
		MULnPO2	0b0

Table 8-8 AONPERIPH Firewall Component regions (continued)

Region number	Peripheral	Field	Value
9	REFCLK CNTBase1	Base Address	0x1A24_0000
		Size	0x0C
		MULnPO2	0b0
10	REFCLK CNTBase2	Base Address	0x1A25_0000
		Size	0x0C
		MULnPO2	0b0
11	REFCLK CNTBase3	Base Address	0x1A26_0000
		Size	0x0C
		MULnPO2	0b0
12	NS WDOG CTRL	Base Address	0x1A30_0000
		Size	0x0C
		MULnPO2	0b0
13	NS WDOG Refresh	Base Address	0x1A31_0000
		Size	0x0C
		MULnPO2	0b0
14	S WDOG CTRL	Base Address	0x1A32_0000
		Size	0x0C
		MULnPO2	0b0
15	S WDOG Refresh	Base Address	0x1A33_0000
		Size	0x0C
		MULnPO2	0b0
16	S32K CNTControl	Base Address	0x1A40_0000
		Size	0x0C
		MULnPO2	0b0
17	S32K CNTRead	Base Address	0x1A41_0000
		Size	0x0C
		MULnPO2	0b0
18	S32K CNTCTL	Base Address	0x1A42_0000
		Size	0x0C
		MULnPO2	0b0
19	S32K CNTBase0	Base Address	0x1A43_0000
		Size	0x0C
		MULnPO2	0b0

Table 8-8 AONPERIPH Firewall Component regions (continued)

Region number	Peripheral	Field	Value
20	S32K CNTBase1	Base Address	0x1A44_0000
		Size	0x0C
		MULnPO2	0b0
21	Interrupt Router	Base Address	0x1A50_0000
		Size	0x0C
		MULnPO2	0b0
22	UART 0	Base Address	0x1A51_0000
		Size	0x0C
		MULnPO2	0b0
23	UART 1	Base Address	0x1A52_0000
		Size	0x0C
		MULnPO2	0b0
24-39	AON Expansion 0-15	Base Address	0x1A60_0000 - 0x1A6F_FFFF, defined by FIREWALL_F0_CFG_SSE700_AONPERIPH_FC_RGN{x}_BASE_ADDR
		Size	0x00, 0x0C-0x14, defined by FIREWALL_F0_CFG_SSE700_AONPERIPH_FC_RGN{x}_SIZE
		MULnPO2	0b0

SYSPERIPH Firewall Component regions

The regions of the SYSPERIPH Firewall Component vary depending on the SSE-700 configuration.

The following table shows the SYSPERIPH Firewall Component regions.

Table 8-9 SYSPERIPH Firewall Component regions

Region number	Peripheral	Field	Value
0	Host System Debug	Base Address	0x1000_0000
		Size	0x1B
		MULnPO2	0b0
1	HSE MHU0	Base Address	0x1B80_0000
		Size	0x0C
		MULnPO2	0b0
2	SEH MHU0	Base Address	0x1B81_0000
		Size	0x0C
		MULnPO2	0b0
3	HSE MHU1	Base Address	0x1B82_0000
		Size	0x0C
		MULnPO2	0b0

Table 8-9 SYSPERIPH Firewall Component regions (continued)

Region number	Peripheral	Field	Value
4	SEH MHU1	Base Address	0x1B83_0000
		Size	0x0C
		MULnPO2	0b0
5	CoreSight SDC-600	Base Address	0x1B90_0000
		Size	0x0C
		MULnPO2	0b0
		Size	0x0C
		MULnPO2	0b0
6	CLUSTOP PPU	Base Address	0x1BC0_0000
		Size	0x0C
		MULnPO2	0b0
7	CORE 0 PPU	Base Address	0x1BC1_0000
		Size	0x0C
		MULnPO2	0b0
8	CORE 1 PPU	Base Address	0x1BC2_0000, only implemented when HOST_CPU_NUM_CORES > 1
		Size	0x0C
		MULnPO2	0b0
9	CORE 2 PPU	Base Address	0x1BC3_0000, only implemented when HOST_CPU_NUM_CORES > 2
		Size	0x0C
		MULnPO2	0b0
10	CORE 3 PPU	Base Address	0x1BC4_0000, only implemented when HOST_CPU_NUM_CORES > 3
		Size	0x0C
		MULnPO2	0b0
7 + HOST_CPU_NUM_CORES	HES0 MHU0	Base Address	0x1B00_0000
		Size	0x0C
		MULnPO2	0b0
8 + HOST_CPU_NUM_CORES	ES0H MHU0	Base Address	0x1B01_0000
		Size	0x0C
		MULnPO2	0b0
9 + HOST_CPU_NUM_CORES	HES0 MHU1	Base Address	0x1B02_0000
		Size	0x0C
		MULnPO2	0b0

Table 8-9 SYSPERIPH Firewall Component regions (continued)

Region number	Peripheral	Field	Value
10 + HOST_CPU_NUM_CORES	ES0H MHU1	Base Address	0x1B03_0000
		Size	0x0C
		MULnPO2	0b0
9 + HOST_CPU_NUM_CORES +	HES1 MHU0	Base Address	0x1B04_0000
		Size	0x0C
		MULnPO2	0b0
10 + HOST_CPU_NUM_CORES +	ES1H MHU0	Base Address	0x1B05_0000
		Size	0x0C
		MULnPO2	0b0
11 + HOST_CPU_NUM_CORES +	HES1 MHU1	Base Address	0x1B06_0000
		Size	0x0C
		MULnPO2	0b0
12 + HOST_CPU_NUM_CORES +	ES1H MHU1	Base Address	0x1B07_0000
		Size	0x0C
		MULnPO2	0b0
7 + HOST_CPU_NUM_CORES + 8	CoreLink GIC-400 Distributor	Base Address	0x1C00_0000
		Size	0x13
		MULnPO2	0b0
7 + HOST_CPU_NUM_CORES + 9	CoreSight STM-500 Extend Stimulus Port	Base Address	0x1D00_0000
		Size	0x18
		MULnPO2	0b0
7 + HOST_CPU_NUM_CORES + 10	NIC(Main) GPV	Base Address	1E00_000
		Size	0x14

DBGPERIPH Firewall Component regions

External debug bus Firewall Component region.

Table 8-10 DBGPERIPH Firewall Component regions

Region Number	Peripheral	Field	Value
0	External Debug Bus	Base Address	0x1800_0000
		Size	0x19
		MULnPO2	0b0

EXPMST0 Firewall Component regions

Depending on the value of the EXPMST0_PE_LVL configuration option the EXPMST0 Firewall Components are described in the following table.

Table 8-11 EXPMST{0-3} Firewall Component Regions

Region number	Peripheral	Field	Value	Notes
The number of regions implemented is set by the EXPMST0_NUM_RGN configuration option.	The peripheral or peripherals protected by the region is IMPLEMENTATION DEFINED.	Base Address	The Base address of the region must be within the address range defined for that Firewall Component. For EXPMST0 Firewall Component: • The Base and Upper Address of a region must be within `0x4000_0000` – `0x5FFF_FFFF`. • The size of the region must be one of the following: `0x0` – Region is disabled; `0x0C` to `0x1C`.	Defined by `FIREWALL_F0_CFG_SSE700_EXPMST0_FC_RGN{x}_BASE_ADDR`
		Size		Defined by `FIREWALL_F0_CFG_SSE700_EXPMST0_FC_RGN{x}_SIZE`

8.4 StreamID and CPUID

Each master or group of masters has a unique StreamID, which is used to identify the transactions issued by that master.

SSE-700 implements a Firewall, as defined in *Appendix C Firewall* on page Appx-C-281. SSE-700 assigns a StreamID value to all the internal masters and provides inputs and outputs on the CVM, XNVM, OCVM, HOSTEXPSLV{0-1}, and HOSTEXPMST{0-1} interfaces.

The width of the inputs and outputs is 8.

Note

The interfaces support the AMBA AXI5 Untranslated_Transaction property including the **AxMMUSID** signals as part of the AR and AW channels. These signals are used to pass the StreamID associated with the transaction.

The following table shows the StreamIDs allocated to masters within SSE-700 and those available for use by the integrator.

Table 8-12 StreamID assignment

StreamID	Master
0	Secure Enclave
1	Host CPU
2	Reserved
3	Host ETR
4	Host AXI-AP
5	SoC ETR
6-15	Reserved
16	External System 0
17	External System 1
18-31	Reserved
32-255	Expansion

The StreamID is passed through SSE-700, unaltered from the slave to master interface, using the **AxMMUSID** signal.

For certain interfaces, as well as the StreamID, there is also a CPUID transported over ARUSER[1:0] and AWUSER[1:0], which identifies which Host CPU core generated the transaction. For information on the interfaces, see *3.1 Interfaces overview* on page 3-31.

The Host CPU core number is binary encoded on the user signals. For transactions generated by another master, those signals are set to one of the following values:

- `0b00` by the Secure Enclave, EXTSYS{0-1}
- Any two-bit value given by HOSTEXPSLV{0-1} through the user signals of the Host Expansion Slave Interface

8.5 Reserved address space and error responses

The reserved address space in an SSE-700 design can be:

- Address space marked as Reserved in Host System address space, defined in *11.1.1 Host System memory map* on page 11-163.

  When any read access (either instruction fetch or data access) is issued to the reserved address space defined in Host System address space, the error response is returned along with a read data from Firewall.

  For internal masters of SSE-700, the read data value is defined as the following table shows. For masters added to SSE-700, for example the External Systems and masters connecting to expansion slave ports, the read data value is defined by the integrator.
- Any unused address space within expansion regions, for example the unused regions of the volatile, non-volatile and off-chip volatile memory regions.

  When any read access is issued to the reserved address space within expansion regions, the response is determined by an integrator. Arm has defined rules for reserved address space the integrator should obey when memory and peripherals are integrated into SSE-700.

The read data response is generated by the Host System Firewall and is configured using the

`FC_ERR_RESP_DEF` or `FIREWALL_F0_CFG_GLOBAL_SSE700_ERR_RESP_PER_MST_ID{x}_VAL`

configuration option. SSE-700 defines the values of the error response read data for masters within SSE-700, including the Host CPU. For masters added to SSE-700, the integrator must set the appropriate value for each master.

The following table shows the value for the error response read data for masters which are part of SSE-700.

Table 8-13 Error response read data for internal masters of SSE-700

Master	Read data value
Secure Enclave	`0xDEAD_DEAD`
Host CPU	`0xB773B773`
Host ETR	`0x0000_0000`
Host AXI-AP	`0x0000_0000`
SoC ETR	`0x0000_0000`

Chapter 9
Host System peripherals

This chapter describes the peripherals, which are part of the SSE-700 Host System.

It contains the following sections:

9.1 Counters and timers

SSE-700 has two time-domains, **REFCLK** and **S32KCLK**.

Both time domains are based on Generic Time, as defined by the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

This section contains the following subsections:

9.1.1 REFCLK time domain

This section describes the **REFCLK** time domain.

The **REFCLK** time domain increments by 1 on each **REFCLK** cycle.

The **REFCLK** time domain includes:

- A memory-mapped counter, **REFCLK** counter. For more information on the **REFCLK** counter register, see *11.3.4 REFCLK Counter CNTControl register summary* on page 11-216
- Four memory-mapped timers, **REFCLK** Timer {0-3} as defined by *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*
- Each Host CPU core implements an Arm Generic Timer, as defined by *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*
- Two Generic Watchdogs, Secure and Non-secure Watchdogs, as defined by the *Arm® Server Base System Architecture 5.0*

The **REFCLK** time domain can be halted during debug, using the Counter CTI.

The **REFCLK** time domain only runs when SSE-700 is in BSYS.RUN or BSYS.SLEEP0 power states. After exiting BSYS.SLEEP1 or BSYS.OFF, software is responsible to restore the counter.

The **REFCLK** time domain is distributed using a 64-bit gray-encoded timestamp value. SSE-700 provides the following interfaces for the **REFCLK** time domain:
- **HOSTCNTVALUEG**: Output from the **REFCLK** counter (gray-encoded timestamp value).
- **HOSTCNTVALUEB**: Input for the **REFCLK** Timers, Secure and Non-secure Watchdogs (binary-encoded timestamp value).

9.1.2 S32KCLK time domain

This section describes the **S32KCLK** time domain.

The **S32KCLK** time domain increments by 1 every S32K cycle and includes:

- A memory-mapped counter, S32K counter, as defined by the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*
- Two memory-mapped timers, S32K Timer 0 and 1, as defined by the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*

The **S32KCLK** time domain can be halted during debug using the Counter CTI.

The **S32KCLK** time domain runs in all SSE-700 power states, except for BSYS.OFF. It is software's responsibility for programming the S32K counter when the SSE-700 exits the BSYS.OFF power state.

The S32KCLK time domain is distributed using a 64-bit gray-encoded timestamp value. SSE-700 provides the following interfaces:
- **HOSTS32KCNTVALUEG**: Output from the S32K counter (gray-encoded timestamp value)
- **HOSTS32KCNTVALUEB**: Input for the S32K Timer 0 and 1 (binary-encoded timestamp value)

9.2 Watchdog

The Host System includes two Generic Watchdogs: Secure and Non-secure.

See *Arm® Server Base System Architecture 5.0* for more information about the Generic Watchdog.

Each Watchdog has a control and refresh frame that is located in the Host System memory map. Both watchdogs use the **REFCLK** timestamp to generate the interrupt events. The two watchdog signals, **WS0** and **WS1**, of the Secure and Non-secure watchdogs are routed as interrupts to:

- Non-secure watchdog **WS0** and **WS1** are both routed to the GIC as interrupts 64 and 33 respectively.
- Secure watchdog **WS0** is routed to the GIC as interrupt 32.
- Secure watchdog **WS1** is routed to the Secure Enclave as interrupt 9.

9.3 Host Base System Control

The Host Base System Control registers allow software to configure features of the Host System:

- Host CPU static configurations
- Host System clock control
- Host CPU power control
- Host CPU boot mask
- Reset Syndrome status
- External System Reset control and status
- Power Control of the Base System

For more information on the Host Base System Control registers, see *11.3.1 Host Base System Control register summary* on page 11-176.

9.4 Interrupt Router

SSE-700 includes an Interrupt Router, which routes the interrupts signals of the EXPSHDINT interface and some internal SSE-700 interrupts between the:

- GICSHDINT interface
- Secure Enclave Cortex-M0+ NVIC
- EXTSYS{0-1}SHDINT interfaces

The Interrupt Router is configured with:

- NUM_ICI between four ICI interfaces:
  - ICI0: Secure Enclave
  - ICI1: Host System GIC
  - ICI2: External System 0
  - ICI3: External System 1
- NUM_SHD_INT is set to 32 plus NUM_EXP_SHD_INT.
- SI{x}_ICI_MSK and SI{x}_DEF_ICI are as defined in the table below.
- Level 2 of the Lockdown Extension
- Secure Enclave is the Security Monitor for the Interrupt Router in SSE-700.

The following table shows the allowed routing for Interrupt Router:

- "Y" indicates that the interrupt can be routed to that ICI{x}
- "N" indicates it cannot be routed to that ICI{x}

Table 9-1 Interrupt Router interface assignment

Interrupt Source	SII	ICI0 (Secure Enclave)	ICI1 (Host System)	ICI{2-3} (External System)
Host System Firewall Interrupt	0	Y[a]	Y	N
SDC-600	1			
Host PPU Combined	2	Y	Y[a]	
REFCLK Timer 0	3			Y
REFCLK Timer 1	4			
REFCLK Timer 2	5			
REFCLK Timer 3	6			
S32K Timer 0	7			
S32K Timer 1	8			
SoC ETR	9	N	Y	
SoC CATU	10			
Reserved	11-31		N	N
EXPSHDINT[n][b]	32+n	-[cd]	-[d]	

[a] Default routing for this interrupt.
[b] When interrupt source is not implemented, this bit of the SII is tied 0 and the SI{x}_ICI_DST is set to 0.
[c] Only the first 32 **EXPSHDINT[n]** can be routed to the Secure Enclave.
[d] Dependent on the value of the SI{x}_ICI_DST and SI{x}_DEF_ICI.

Note

For **EXPSHDINT[n]**, n is between 0 and NUM_EXP_SHD_INT.

9.4.1 IMPLEMENTATION DEFINED behavior

This section describes the IMPLEMENTATION DEFINED behaviors that are visible or impact on the SSE-700.

Note

This section must be read with the *Appendix C Firewall* on page Appx-C-281.

- Both level and edge-based shared interrupts are supported.
- Lockdown interface is implemented.
- Tamper Interrupt interface is implemented. Tamper Interrupt is reported to Secure Enclave.

Note

While the Interrupt Router supports both level and edge-based interrupts, the destination of that interrupt must be considered. For example, the Secure Enclave interrupts only support level-based interrupts.

Appendix B Interrupt Router on page Appx-B-274 defines the generic behavior of the Interrupt Router.

9.5 MHU

SSE-700 provides *Message Handling Units* (MHUs).

The MHUs are compliant to the MHUv2.1 specification, defined in *Appendix A Message Handling Unit* on page Appx-A-252.

An MHU has two halves:

- A Sender frame, or Sender
- A Receiver frame, or Receiver

The MHU facilitates communication between two systems in SSE-700, with one system having the Sender and the other having the Receiver of a single MHU.

MHUs are implemented in pairs to allow for full-duplex communications, with Sender and Receiver being reversed between the two MHUs in the pair. For example, between the Host System and Secure Enclave: one of the MHUs has the Sender in the Host System and the Receiver in the Secure Enclave, while the other MHU has the Sender in the Secure Enclave and the Receiver in the Host System.

SSE-700 Subsystem supports Arm TrustZone® for each External System, two pairs of MHUs are implemented. This enables software to use one pair for Secure communication and the other for Non-secure communication.

Note

In this document, the MHUs are either referred to by the MHU Name or Short Name, defined in the table below. When referring to either the Sender or Receiver frame, the Short Name is followed by either Sender or Receiver.

The MHU is designed to be split across clock, reset, and power boundaries. The software is responsible for requesting the Receiver portion of the MHU is powered. The software does this by using the Ready to Send protocol defined in *Appendix A Message Handling Unit* on page Appx-A-252. Software is also responsible for making sure the Sender frame remains powered until the message has been received by the Receiver. How software knows when the Receiver has received the message, depends on the Transport Protocol being used. For more information on Transport Protocols see *A.3 Transport protocols* on page Appx-A-270.

The following table shows the SSE-700 MHUs.

Table 9-2 MHUs

MHU name	Short name	Sender system	Receiver system
Host to Secure Enclave MHU0	HSE0	Host	Secure Enclave
Secure Enclave to Host MHU0	SEH0	Secure Enclave	Host
Host to Secure Enclave MHU1	HSE1	Host	Secure Enclave
Secure Enclave to Host MHU1	SEH1	Secure Enclave	Host
Secure Enclave to External System 0 MHU 0	SEES00	Secure Enclave	External System 0
External System 0 to Secure Enclave MHU 0	ES0SE0	External System 0	Secure Enclave
Secure Enclave to External System 0 MHU 1	SEES01	Secure Enclave	External System 0
External System 0 to Secure Enclave MHU 1	ES0SE1	External System 0	Secure Enclave
Secure Enclave to External System 1 MHU 0	SEES10	Secure Enclave	External System 1
External System 1 to Secure Enclave MHU 0	ES1SE0	External System 1	Secure Enclave
Secure Enclave to External System 1 MHU 1	SEES11	Secure Enclave	External System 1

Table 9-2 MHUs (continued)

MHU name	Short name	Sender system	Receiver system
External System 1 to Secure Enclave MHU 1	ES1SE1	External System 1	Secure Enclave
Host to External System 0 MHU 0	HES00	Host	External System 0
External System 0 to Host MHU 0	ES0H0	External System 0	Host
Host to External System 0 MHU 1	HES01	Host	External System 0
External System 0 to Host MHU 1	ES0H1	External System 0	Host
Host to External System 1 MHU 0	HES10	Host	External System 1
External System 1 to Host MHU 0	ES1H0	External System 1	Host
Host to External System 1 MHU 1	HES11	Host	External System 1
External System 1 to Host MHU 1	ES1H1	External System 1	Host

Using the MHU_{x}_NUM_CH parameter (where x is the MHU short name), all MHUs in SSE-700, can be configured with 1 to 32 channels.

9.6 Boot Register

The Boot Register is a write-once set of registers providing the initial instructions to the Host CPU.

The Boot Register is in the AONTOP power domain and uses **REFCLK**. This enables the Boot Register to retain its values in all power states.

The Boot Register is only accessible through:

- Secure read
- Secure writes from the Secure Enclave only

Any other accesses generate an error response.

The method by which the Boot Register identifies the master which issued the read or write operation, uses the StreamID. For more information, see *8.4 StreamID and CPUID* on page 8-144.

For more information on the register assignment of the Boot Register, see *11.3.6 Boot register aummary* on page 11-224.

9.7 CoreSight SDC-600

To enable a debug agent to pass data to the target system, for example the passing of a debug authentication certificate, SSE-700 includes a CoreSight SDC-600 component.

The CoreSight SDC-600 in SSE-700 has two parts:

- An External APBCOM, compliant with the ADIv6 specification, in the AONTOP power domain and in the **REFCLK** domain. It is connected to the External Debug Bus.
- An Internal APBCOM in the SYSTOP power domain and in the **ACLK** domain, connected to the NIC. The interrupt is routed, via the Interrupt Router, to the Host System GIC or the NVIC of the Secure Enclave.

Note

- In this manual, the External and Internal APBCOM are referred to as EXT APBCOM and INT APBCOM respectively.
- SSE-700 does not support the usage of the REMRR to generate a reset request and the debug agent must use the **nSRST** or the CSYSRSTREQ of the DP ROM.

For more information on the CoreSight SDC-600, see the *Arm® CoreSight™ SDC-600 Secure Debug Channel Technical Reference Manual*.

9.8 UART

SSE-700 includes two PL011 UARTs} in the AONTOP domain, Host UART{0,1}.

The UART supports the hardware-based follow control. Host UART{0,1} uses the HOSTUARTCLK for both the PCLK and UARTCLK inputs of PL011. The Host System only uses the combined interrupt from the UART.

For more information on the UARTs, see the *PrimeCell UART (PL011) Technical Reference Manual*.

Note

SSE-700 only supports the use of the PL011 as an RS232-compliant UART.

Chapter 10
Boot

This chapter describes the boot requirements and flow of the SSE-700.

It contains the following sections:

10.1 Boot overview

When **PORESETn** is released on SSE-700, the Secure Enclave is the only system that initially boots.

Both the Host and External Systems are prevented from executing any instructions, but access to resources within the SSE-700 is allowed. For example, the Secure Enclave can access the counters in the Host System to initialize the time domains.

SSE-700 does not mandate a specific boot flow but certain steps are required to be completed during the boot process.

10.2 Boot requirements on page 10-159 defines the boot requirements for SSE-700. *10.3 Example boot flow* on page 10-160 shows an example boot flow, for reference only.

10.2 Boot requirements

This section describes actions that must be performed during the boot process. Certain steps can be performed by more than one system.

The following table shows the actions required as part of the boot process. The order of the rows does not indicate any ordering in the steps required.

Table 10-1 Boot requirements

Step	Secure Enclave	Host System	External System
Configure the Secure Enclave, including: • Enable the SoC and Secure Enclave watchdogs • Configure the Secure Enclave Firewall, to allow access to the Host System	Y	N	N
Initialize Host System Firewall: program the Host System Firewall to gain access to the CVM, OCVM, and NVM			
Host System Firewall full programming: program the Host System Firewall for all masters and slaves	Y[a]	Y[a]	
Initialize S32K counter	Y[b]	Y[b]	
Initialize REFCLK counter	Y[bc]	Y[bc]	
Initialize NVM : configure the non-volatile memory, the Flash Controller	Y	Y	
Load Secure Enclave firmware into Secure Enclave RAM and authenticate		N	
Load Host System Secure firmware into volatile memory and authenticate			
Authenticate Host System Non-secure firmware and OS			
Authenticate External System firmware[d]			
Write to the Host System Boot Register			
Release the Host CPU: • Clear the CPUWAIT in the Secure Enclave Base System Control registers • Wake the respective Host CPU core by performing an action which will cause the Host CPU Core power domain to exit the OFF power mode			
Release the External System: clear the CPUWAIT, for the External System, in the Host System Base System Control registers	Y[e]	Y[e]	

[a] Full programming of the Host System Firewall can be done by either the Host CPU or Secure Enclave, depending on the system requirements.
[b] Initializing the S32K and REFCLK counters is only required to be done by one system.
[c] The REFCLK counter will need to be re-initialized every time SSE-700 exits the BSYS.SLEEP1 or BSYS.OFF power states.
[d] If the External System firmware is stored in a separate non-volatile storage device to the one connected to the NVM interface, the Secure Enclave must be able to access the storage device to authenticate the image.
[e] The External System can be released by either Secure Enclave or Host CPU, but only once the External System firmware has been authenticated.

10.3 Example boot flow

This section contains an example boot flow for reference only.

1. The Secure Enclave is released from reset and starts executing from the Secure Enclave ROM.
2. This ROM code configures the Secure Enclave system, which includes programming the Secure Enclave Firewall to gain access to the Host System.
3. Secure Enclave performs initial programming of the Host System Firewall. This includes:
   a. Granting access to the volatile and non-volatile memory for both the Host CPU and Secure Enclave. At this stage Arm recommends that only Secure access is granted.
   b. Granting access to any peripherals which are required. For example, if the Secure Watchdog is to be enabled, grant access to the REFCLK counter and Secure Watchdog.

   Note

   For the Secure Watchdog to be used, REFCLK counter must be initialized beforehand.

4. The Secure Enclave configures access to the Non-volatile memory, for example the Flash Controller.
5. The Secure Enclave loads and authenticates its own firmware into the Secure Enclave RAM. Decryption can be applied at this point.

   Note

   To avoid time-of-check to time-of-use attacks, the Secure Enclave must load the firmware image into the Secure Enclave RAM before performing the authentication. This task can be processed in parallel to reduce the time taken.

6. If the authentication succeeds, the Secure Enclave starts to execute from the loaded image. If authentication fails, a backup image is used if available, or it enters a recovery mode.
7. The Secure Enclave loads the Host CPU Secure firmware into the on-chip volatile memory and authenticates it. No Non-secure firmware is required to be loaded at this point. Decryption can be applied at this stage. If authentication fails, the Secure Enclave can attempt to authenticate the backup image, if available, or enter a recovery mode.
8. The Secure Enclave writes the Boot Register with data values of instruction opcodes to cause the Host CPU to branch into the correct location in its Secure firmware stored in on-chip volatile memory.
9. The Secure Enclave sets HOST_SYS_RST_CTRL.CPUWAIT to `0b0` to allow the Host CPU processors to execute instructions.
10. The Secure Enclave writes `0b1` to HOST_CPU_WAKEUP.CORE0_WAKEUP to request Core 0 to start executing instructions from the Boot Register.

    Note

    In this example, Core 0 is the Core selected to perform the initial boot but any implemented Core or multiple Cores could be used.

11. The Host CPU writes `0b0` to the HOST_CPU_WAKEUP.CORE0_WAKEUP to remove the wakeup request for Core 0.
12. The Host CPU Secure firmware configures the system. For example, the clock and power settings for the SoC.
13. At this stage, the full programming of the Firewall must be performed. This can be done by either the Host CPU Secure firmware or the Secure Enclave.
14. The Host CPU Secure firmware requests authentication of the Non-secure firmware and Rich OS from the Secure Enclave. If authentication fails the Host CPU can request authentication of a backup image, if available, or enter a recovery mode.
15. The Host CPU boots the Non-secure firmware and Rich OS. It is IMPLEMENTATION DEFINED, whether the Non-secure firmware and Rich OS are:

a. Loaded into the On-chip volatile memory.
b. Loaded into the Off-chip volatile memory.
c. Executed in-place.

16. The Rich OS requests authentication of the External System firmware from the Secure Enclave. If required, any decryption or transferring to a local memory within the External System can be performed. If authentication fails, the Rich OS can request authentication of a backup image if available, or enter a recovery mode.
17. The Rich OS sets the EXT_SYS{0-1}_RST_CTRL.CPUWAIT to `0b0` to allow the External System processors to start to execute instructions.

Note

If there is more than one External System, steps 16 and 17 must be performed for each External System. It is IMPLEMENTATION DEFINED whether the Rich OS performs this in series or in parallel.

Chapter 11
Programmers model

This chapter describes the SSE-700 memory and interrupt maps, and provides detailed information for all programmable registers.

It contains the following sections:

11.1 Memory map

This section describes the SSE-700 memory maps.

This section contains the following subsections:

11.1.1 Host System memory map

Peripherals and devices in the Host System are allocated memory in units of 64KB pages. This allows systems to use any granular page size defined by the table page formats.

Any unoccupied regions of its allocation are Reserved. Reserved regions are treated as RAZ/WI and generate an error if accessed.

Some regions of memory are marked as Secure. Secure access can access these regions, while Non-Secure access is blocked and generates an error.

- Issuing master
- Security
- Privilege type
- Access type (Read/Write/Execute)

The Firewall applies additional constraints, and it cannot remove any constraints already specified. A secure access region cannot be made accessible to non-secure access attempts.

At the highest level, the Host System Memory Map is divided into a few key regions as the table below shows.

Note

In the following table the omission of the Security column, or an entry of "-", indicates that the region or peripheral is accessed from any security world.

The entire Host System memory map is protected by the Host System Firewall, see *Appendix C Firewall* on page Appx-C-281. It enables software to limit access to areas of the memory map, based on the following properties:

Table 11-1 Host System memory map

Offset	Size	Security	Region	Notes
0x00_0000_0000	4KB	Secure	Boot Register	*11.3.6 Boot register aummary* on page 11-224
0x00_0000_1000	1020KB	-	Reserved	-
0x00_0010_0000	15MB	-	Reserved	-
0x00_0100_0000	16MB	-	Reserved	-
0x00_0200_0000	32MB	-	Volatile Memory	*3.4.1 On-chip Volatile Memory (CVM) interface* on page 3-46
0x00_0400_0000	64MB	-	Reserved	-

Table 11-1 Host System memory map (continued)

Offset	Size	Security	Region	Notes
0x00_0800_0000	128MB	-	eXecute-in-place Non-Volatile Memory	*3.4.2 eXecute-in-place Non-volatile Memory (XNVM) interface* on page 3-46
0x00_1000_0000	160MB	-	Debug	*Debug region* on page 11-164
0x00_1A00_0000	608MB	-	Host Peripherals	*Host Peripheral Region* on page 11-165
0x00_4000_0000	1GB	-	Host Master Expansion	*Host Master Expansion Region* on page 11-168
0x00_8000_0000	2GB	-	Off-chip Volatile Memory	*3.4.3 Off-chip Volatile Memory (OCVM) interface* on page 3-47
0x01_0000_0000	1020GB	-	Reserved	-

Debug region

The following table shows the Debug region of the Host System memory map.

Table 11-2 Debug region memory map

Offset	Size	Component	Notes
0x00_1000_0000	128MB	Host System Debug	See *Host System Debug* on page 11-164
0x00_1800_0000	32MB	External Debug Bus	See *11.1.2 External Debug Bus memory map* on page 11-169

Note

The Host Debug area provides access to the debug infrastructure of the Host System. The External Debug Bus region provides access to all debug logic, for example, the debug infrastructure of an External System.

Host System Debug

The Host System Debug area provides access to the debug infrastructure of the Host System.

This memory is accessed via the:

- Host System processor and Secure Enclave, via the Host System memory map at 0x1000_0000.
- Any External Debug Bus master, using the Host APB AP in the External Debug Memory Map, for example, the DP. See *11.1.2 External Debug Bus memory map* on page 11-169.

The following table shows the Host Debug memory map.

Table 11-3 Host Debug memory map

Offset	Size	Component	Notes
0x0000_0000	64KB	Host Debug ROM	-
0x0001_0000	576KB	Reserved	-
0x000A_0000	64KB	Host Funnel	-

Table 11-3 Host Debug memory map (continued)

Offset	Size	Component	Notes
0x000B_0000	384KB	Reserved	-
0x0011_0000	64KB	Host Replicator	-
0x0012_0000	64KB	Host ETR	-
0x0013_0000	64KB	Host CATU	-
0x0014_0000	64KB	Host CTI	-
0x0015_0000	128KB	Reserved	-
0x0017_0000	64KB	STM APB	-
0x0018_0000	512KB	Reserved	-
0x0020_0000	14MB	Reserved	-
0x0100_0000	16MB	Expansion	Maps to Host Debug APB expansion Interface (HOSTDBGEXP)
0x0200_0000	16MB	Host CPU Debug	Maps to Host CPU Debug APB Interface (HOSTCPUDBG)
0x0300_0000	16MB	Host CPU Debug Internal View	Maps to Host CPU Debug APB Interface (HOSTCPUDBG) with the **PADDR31** signal tied LOW
0x0400_0000	64MB	Reserved	-
0x0800_0000	896MB	Reserved	Not directly accessible in the Host System memory map
0x4000_0000	3GB	Reserved	Not directly accessible in the Host System memory map

Host Peripheral Region

The Host Peripheral Region gives access to all peripherals within the Host System.

This memory is accessed via the system interconnect by the:

- Host Processor
- Secure Enclave
- Host Slave Expansion interfaces (HOSTEXPSLV{0-1})
- External Systems

The following table shows the Host Peripheral Region peripherals and memory area.

Table 11-4 Host Peripheral Memory Map

Offset	Size	Security	Component	Notes
0x1A00_0000	64KB	-	System ID	See *11.3.6 Boot register aummary* on page 11-224
0x1A01_0000	64KB	-	Host Base System Control	See *11.1.1 Host System memory map* on page 11-163
0x1A02_0000	64KB	Secure	Firewall PPU	See *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual*

Table 11-4 Host Peripheral Memory Map (continued)

Offset	Size	Security	Component	Notes
0x1A03_0000	64KB	Secure	SYSTOP PPU	See *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual*
0x1A04_0000	64KB	Secure	DBGTOP PPU	See *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual*
0x1A05_0000	1728KB	-	Reserved	-
0x1A20_0000	64KB	Secure	REFCLK CNTControl	See *9.1 Counters and timers* on page 9-147
0x1A21_0000	64KB	-	REFCLK CNTRead	
0x1A22_0000	64KB	-	REFCLK CNTCTL	
0x1A23_0000	64KB	-	REFCLK CNTBase0	
0x1A24_0000	64KB	-	REFCLK CNTBase1	
0x1A25_0000	64KB	-	REFCLK CNTBase2	
0x1A26_0000	64KB	-	REFCLK CNTBase3	
0x1A27_0000	640KB	-	Reserved	
0x1A30_0000	64KB	-	NS WDOG CTRL	
0x1A31_0000	64KB	-	NS WDOG Refresh	
0x1A32_0000	64KB	Secure	Secure WDOG CTRL	
0x1A33_0000	64KB	Secure	Secure WDOG Refresh	
0x1A34_0000	768KB	-	Reserved	
0x1A40_0000	64KB	Secure	S32K CNTControl	
0x1A41_0000	64KB	-	S32K CNTRead	
0x1A42_0000	64KB	-	S32K CNTCTL	
0x1A43_0000	64KB	-	S32K CNTBase0	
0x1A44_0000	64KB	-	S32K CNTBase1	
0x1A45_0000	704KB	-	Reserved	-
0x1A50_0000	64KB	-	Interrupt Router	See *Appendix B Interrupt Router* on page Appx-B-274
0x1A51_0000	64KB	-	UART0	See *9.8 UART* on page 9-156, *Appendix B Interrupt Router* on page Appx-B-274
0x1A52_0000	64KB	-	UART1	See *9.8 UART* on page 9-156, *Appendix B Interrupt Router* on page Appx-B-274
0x1A53_0000	832KB	-	Reserved	-
0x1A60_0000	1MB	-	AON Expansion	-
0x1A70_0000	1MB	-	Reserved	-
0x1A80_0000	2MB	-	Host System Firewall	See *8.3.5 Host System Firewall* on page 8-134
0x1AA0_0000	6MB	-	Reserved	-

Table 11-4 Host Peripheral Memory Map (continued)

Offset	Size	Security	Component	Notes
0x1B00_0000	64KB	-	Host to External System 0 MHU0	See *9.5 MHU* on page 9-152
0x1B01_0000	64KB	-	External System 0 to Host MHU0	
0x1B02_0000	64KB	-	Host to External System 0 MHU1	See *9.5 MHU* on page 9-152
0x1B03_0000	64KB	-	External System 0 to Host MHU1	
0x1B04_0000	64KB	-	Host to External System 1 MHU0	See *9.5 MHU* on page 9-152
0x1B05_0000	64KB	-	External System 1 to Host MHU0	
0x1B06_0000	64KB	-	Host to External System 1 MHU1	See *9.5 MHU* on page 9-152
0x1B07_0000	64KB	-	External System 1 to Host MHU1	
0x1B08_0000	512KB	-	Reserved	-
0x1B10_0000	7MB	-	Reserved	-
0x1B80_0000	64KB	-	Host to Secure Enclave MHU0	See *9.5 MHU* on page 9-152
0x1B81_0000	64KB	-	Secure Enclave to Host MHU0	
0x1B82_0000	64KB	-	Host to Secure Enclave MHU1	
0x1B83_0000	64KB	-	Secure Enclave to Host MHU1	
0x1B84_0000	768KB	-	Reserved	-
0x1B90_0000	64KB	-	INT APBCOM	See *9.7 CoreSight SDC-600* on page 9-155
0x1B91_0000	64KB	-	Reserved	-
0x1B92_0000	64KB	-	Reserved	-
0x1B93_0000	832KB	-	Reserved	-
0x1BA0_0000	6MB	-	Reserved	-
0x1BC0_0000	64KB	Secure	CLUSTOP PPU	See *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual*
0x1BC1_0000	64KB	Secure	CORE 0 PPU	See *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual*
0x1BC2_0000	64KB	Secure	CORE 1 PPU	Reserved if HOST_NUM_CPU_CORE < 2 See *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual*

Table 11-4 Host Peripheral Memory Map (continued)

Offset	Size	Security	Component	Notes
0x1BC3_0000	64KB	Secure	CORE 2 PPU	Reserved if HOST_NUM_CPU_CORE < 3 See *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual*
0x1BC4_0000	64KB	Secure	CORE 3 PPU	Reserved if HOST_NUM_CPU_CORE < 4. See *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual*
0x1BC5_0000	3778KB	-	Reserved	-
0x1C00_0000	16MB	-	GIC Region	See *GIC Region* on page 11-168
0x1D00_0000	16MB	-	CoreSight STM-500 Extended Stimulus Port	See *7.6 System Trace Macrocell* on page 7-116
0x1E00_0000	1MB	Secure	NIC GPV	See *Arm® CoreLink™ NIC-450 Network Interconnect Technical Overview*
0x1E10_0000	543MB	-	Reserved	-

GIC Region

SSE-700 subsystem supports GIC-400.

The following table shows the GIC-400 memory region allocation.

Table 11-5 GIC region allocation

Offset	Size	Component
0x000_0000	64KB	Reserved
0x001_0000	4KB	GIC Distributor
0x001_1000	120KB	Reserved
0x002_F000	8KB	GIC CPU Interface
0x003_1000	120KB	Reserved
0x004_F000	4KB	GIC Virtual Interface Control
0x005_0000	4KB	GIC Virtual Interface Control Alias
0x005_1000	120KB	Reserved
0x006_F000	8KB	GIC Virtual CPU Interface
0x007_1000	15932KB	Reserved

Note

For more information on the programmers model of GIC see the *CoreLink™ GIC-400 Generic Interrupt Controller Technical Reference Manual*.

Host Master Expansion Region

The Host Master Expansion Region contains a 1GB area of memory routed to Host Expansion Master {0-1} (HOSTEXPMST{0-1}) Interfaces. Each interface is assigned 512MB.

The table below shows the Host Master Expansion Region.

Table 11-6 Host Master Expansion Region

Offset	Size	Security	Component	Notes
0x00_4000_0000	512MB	-	EXPMST0	-
0x00_6000_0000	512MB	-	EXPMST1	-

By adding use-case specific peripherals the integrator can expand the SSE-700 subsystem.

Access to the Host Expansion Master {0-1} interfaces are filtered by the EXPMST{0-1} Firewall Components. For more information on the Host System Firewall, see *8.3.5 Host System Firewall* on page 8-134.

11.1.2 External Debug Bus memory map

The External Debug Bus memory map provides access to all debug logic in the SSE-700 subsystem, including the SoC Debug infrastructure.

Access is via different masters using different offsets:

- Host System processor and Secure Enclave, via the Host System memory map, at 0x1800_0000
- JTAG/Serial Wire Debug, via the DP at 0x0000_0000
- Each associated External System External Debug Interface (EXTSYS{0-1}EXTDBG) of the External System, at an IMPLEMENTATION DEFINED address

The table below shows the External Debug Bus memory map.

External Debug Memory Map access is controlled via the Debug Authorization Access Control Gate.

Table 11-7 External Debug Bus memory map

Offset	Size	Component	Notes
0x0000_0000	64KB	DP ROM	Only accessible by the DP. When accessed from any other master this location is Reserved.
0x0001_0000	64KB	GPIO Control	
0x0002_0000	64KB	EXT APBCOM	-
0x0003_0000	64KB	EXTDBG ROM	-
0x0004_0000	64KB	Secure Enclave AP	-
0x0005_0000	64KB	Host APB AP	-
0x0006_0000	64KB	Host AXIAP ROM	-
0x0007_0000	64KB	Host AXI AP	-
0x0008_0000	512KB	Reserved	-
0x0010_0000	64KB	SoC TPIU Funnel	-
0x0011_0000	64KB	SoC TPIU Replicator	-
0x0012_0000	64KB	SoC TPIU	-
0x0013_0000	64KB	SoC CTI	-
0x0014_0000	64KB	SoC ETR	-
0x0015_0000	64KB	SoC CATU	-
0x0016_0000	64KB	Counter CTI	-

Table 11-7 External Debug Bus memory map (continued)

Offset	Size	Component	Notes
0x0017_0000	576KB	Reserved	-
0x0020_0000	1MB	External System 0 Debug	-
0x0030_0000	1MB	External System 1 Debug	-
0x0040_0000	1MB	Reserved	-
0x0050_0000	1MB	Reserved	-
0x0060_0000	1018MB	Reserved	-
0x4000_0000	3GB	Reserved	-

Note

Reserved locations on the External Debug Bus are treated as RAZ/WI and generate an error when accessed.

External System {0-1} Debug

The External System{0-1} Debug regions are provided to allow the External System to implement its own debug logic.

Arm recommends using only CoreSight ROM tables and APs in the External System {0-1} Debug regions. Accessing unused address space returns an error.

11.1.3 Secure Enclave Memory Map

Secure Enclave has a 32-bit Address space.

See the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual* for details of the Secure Enclave memory map.

11.1.4 External System memory map

The EXTSYS{0-1}MHU interface permits access to the MHUs. The following table shows the External System memory map.

Note

The full memory map of the External System is IMPLEMENTATION DEFINED.

Table 11-8 Memory map of External System MHU interface

Offset	Size	Security	Component	Notes
0x0_0000	4KB	Secure	HES{x}0 Receiver	-
0x0_1000	60KB	-	Reserved	RAZ/WI and slave error is returned
0x1_0000	4KB	Secure	ES{x}H0 Sender	-
0x1_1000	60KB	-	Reserved	RAZ/WI and slave error is returned
0x2_0000	4KB	-	HES{x}1 Receiver	-
0x2_1000	60KB	-	Reserved	RAZ/WI and slave error is returned
0x3_0000	4KB	-	ES{x}H1 Sender	-

Table 11-8 Memory map of External System MHU interface (continued)

Offset	Size	Security	Component	Notes
0x3_1000	60KB	-	Reserved	RAZ/WI and slave error is returned
0x4_0000	4KB	Secure	SEES{x}0 Receiver	-
0x4_1000	60KB	-	Reserved	RAZ/WI and slave error is returned
0x5_0000	4KB	Secure	ES{x}SE0 Sender	-
0x5_1000	60KB	-	Reserved	RAZ/WI and slave error is returned
0x6_0000	4KB	-	SEES{x}1 Receiver	-
0x6_1000	60KB	-	Reserved	RAZ/WI and slave error is returned
0x7_0000	4KB	-	ES{x}SE1 Sender	-
0x7_1000	60KB	-	Reserved	RAZ/WI and slave error is returned

Note

{x} is the External System number (0-1).

11.2 Interrupt map

This section covers only the interrupt map of the Host Processor, for details on the Secure Enclave interrupt map see the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

This section contains the following subsection:

11.2.1 Host CPU interrupt map

The Host processor interrupt map depends on the configuration of the SSE-700.

Table 11-9 Host CPU interrupt map

Interrupt number	Interrupt source	Type	Level/edge	Notes
0-15	Reserved	SGI	Edge	-
16-24	Reserved	PPI	-	-
25	Virtual CPU Interface Maintenance	PPI	Level	-
26	Hypervisor timer	PPI	Level	-
27	Virtual timer	PPI	Level	-
28	Reserved	PPI	Level	-
29	Secure physical timer	PPI	Level	-
30	Non-secure physical timer	PPI	Level	-
31	Reserved	PPI	-	-
32	Secure WDOG WS0	SPI	Level	-
33	Non-secure WDOG WS1	SPI	Level	-
34	**REFCLK** Timer 0	SPI	Level	-
35	S32K Timer 0	SPI	Level	-
36	Host System Firewall	SPI	Level	-
37	Host PPU Combined	SPI	Level	-
38	SDC-600	SPI	Level	-
39	CPU **nEXTERRIRQ**	SPI	Level	See *3.2 Host CPU-GIC socket interfaces* on page 3-32 for more information
40	CPU **nINTERRIRQ**	SPI	Level	
41	Host to Secure Enclave MHU0 Sender Combined IRQ	SPI	Level	-
42	Secure Enclave to Host MHU0 Receiver Combined IRQ	SPI	Level	-

Table 11-9 Host CPU interrupt map (continued)

Interrupt number	Interrupt source	Type	Level/edge	Notes
43	Host to EXTSYS0 MHU0 Sender Combined IRQ	SPI	Level	-
44	EXTSYS 0 to Host MHU0 Receiver Combined IRQ	SPI	Level	-
45	Host to EXTSYS 1 MHU0 Sender Combined IRQ	SPI	Level	-
46	EXTSYS 1 to Host MHU0 Receiver Combined IRQ	SPI	Level	-
47-50	Reserved	SPI	-	-
51	Host UART0 UARTINTR	SPI	Level	-
52	Host UART1 UARTINTR	SPI	Level	-
53-63	SoC Expansion	SPI	IMPLEMENTATION DEFINED	The interrupts could be from GICSHDINT or the dedicated interrupts which do not pass through the Interrupt Router
64	Non-secure WDOG WS0	SPI	Level	-
65	**REFCLK** Timer 1	SPI	Level	-
66	**REFCLK** Timer 2	SPI	Level	-
67	**REFCLK** Timer 3	SPI	Level	-
68	S32K Timer 1	SPI	Level	-
69	Host STM Sync IRQ	SPI	Edge	-
70	Host ETR Buffer IRQ	SPI	Level	-
71	Host CATU IRQ	SPI	Level	-
72	Host CTI Trigger Out 4	SPI	Edge	-
73	Host CTI Trigger Out 5	SPI	Edge	-
74	SoC ETR IRQ	SPI	Level	-
75	SoC CATU IRQ	SPI	Level	-
76	Host to Secure Enclave MHU1 Sender Combined IRQ	SPI	Level	-
77	Secure Enclave to Host MHU1 Receiver Combined IRQ	SPI	Level	-

Table 11-9 Host CPU interrupt map (continued)

Interrupt number	Interrupt source	Type	Level/edge	Notes
78	Host to EXTSYS 0 MHU1 Sender Combined IRQ	SPI	Level	-
79	EXTSYS 0 to Host MHU1 Receiver Combined IRQ	SPI	Level	-
80	Host to EXTSYS 1 MHU1 Sender Combined IRQ	SPI	Level	-
81	EXTSYS 1 to Host MHU1 Receiver Combined IRQ	SPI	Level	-
82-85	Reserved	SPI	-	-
86	Host CPU0 Debug Comm TX Full/ Host CPU0 Debug Comm Channel	SPI	Level	-
87	Reserved	SPI	-	-
88	Host CPU0 PMU Counter Overflow	SPI	Level	-
89	Host CPU0 CTI Trigger	SPI	Edge	-
90	Reserved	SPI	-	-
91	Host CPU1 Debug Comm TX Full/ Host CPU1 Debug Comm Channel	SPI	Level	Only implemented when HOST_CPU_NUM_CORES > 1. Otherwise Reserved.
92	Reserved	SPI	-	-
93	Host CPU1 PMU Counter Overflow	SPI	Level	Only implemented when HOST_CPU_NUM_CORES > 1. Otherwise Reserved.
94	Host CPU1 CTI Trigger	SPI	Edge	
95	Reserved	SPI	-	-
96	Host CPU2 Debug Comm TX Full/ Host CPU2 Debug Comm Channel	SPI	Level	Only implemented when HOST_CPU_NUM_CORES > 2. Otherwise Reserved.
97	Reserved	SPI	-	-
98	Host CPU2 PMU Counter Overflow	SPI	Level	Only implemented when HOST_CPU_NUM_CORES > 2. Otherwise Reserved.
99	Host CPU2 CTI Trigger	SPI	Edge	
100	Reserved	SPI	-	-
101	Host CPU3 Debug Comm TX Full/ Host CPU3 Debug Comm Channel	SPI	Level	Only implemented when HOST_CPU_NUM_CORES > 3. Otherwise Reserved.

Table 11-9 Host CPU interrupt map (continued)

Interrupt number	Interrupt source	Type	Level/edge	Notes
102	Reserved	SPI	-	-
103	Host CPU3 PMU Counter Overflow	SPI	Level	Only implemented when HOST_CPU_NUM_CORES > 3. Otherwise Reserved.
104	Host CPU3 CTI Trigger	SPI	Edge	Only implemented HOST_CPU_NUM_CORES > 3. Otherwise Reserved.
105-127	Reserved	SPI	-	-
>=128	SoC Expansion	SPI	IMPLEMENTATION DEFINED	The interrupts could be from GICSHDINT or the dedicated interrupts which do not pass through the Interrupt Router

Note

Reserved interrupts are tied LOW when not implemented.

Interrupt IDs 32-63 are lockable when they are configured as secure interrupts. For more information, see *Arm® Generic Interrupt Controller Architecture Specification, architecture version 2.0*.

11.3 Register descriptions

This chapter describes the registers for SSE-700 components.

For the following standalone components, see their individual references:

- PPU
- Coresight IP components
- SDC-600
- Cortex-A32
- NIC-400 GPV
- UART

For detailed register descriptions of the Firewall and MHU, see the Firewall and MHU sections in the Appendices of this document.

This section contains the following subsections:

11.3.1 Host Base System Control register summary

This section summarizes the Host Base System Control registers.

The Host Base System Control has registers to control the clocks, power, and reset for SSE-700. It resides within AONTOP power domain. The registers can be accessed at offset `0x1A01_0000` in the Host System memory map.

Table 11-10 Host Base System Control registers summary

Offset	Short name	Access	Name
`0x000`	CLUSTER_CONFIG	RW	Cluster Static Config *Cluster Static Config (CLUS_CFG) register* on page 11-180
`0x004` – `0x00C`	-	RO	Reserved
`0x010`	PE0_CONFIG	RW	Processing Element 0 Static Config *Processing Element {0-3} Static Config (PE{0-3}_CFG) register* on page 11-180
`0x014`	PE0_RVBARADDR_LW	RW	Processing Element 0 Reset Vector Base Address Lower *Processing Element {0-3} Reset Vector Base Address Lower and Upper (PE{0-3}_RVBAR_LW) register* on page 11-181
`0x018`	PE0_RVBARADDR_UP	RO	Processing Element 0 Reset Vector Base Address Upper *Processing Element {0-3} Reset Vector Base Address Upper (PE{0-3}_RVBAR_UP) register* on page 11-182
`0x01C`	-	RO	Reserved
`0x020`	PE1_CONFIG	RW	Processing Element 1 Static Config *Processing Element {0-3} Static Config (PE{0-3}_CFG) register* on page 11-180
`0x024`	PE1_RVBARADDR_LW	RW	Processing Element 1 Reset Vector Base Address Lower *Processing Element {0-3} Reset Vector Base Address Lower and Upper (PE{0-3}_RVBAR_LW) register* on page 11-181

Table 11-10 Host Base System Control registers summary (continued)

Offset	Short name	Access	Name
0x028	PE1_RVBARADDR_UP	RO	Processing Element 1 Reset Vector Base Address Upper *Processing Element {0-3} Reset Vector Base Address Upper (PE{0-3}_RVBAR_UP) register* on page 11-182
0x02C	-	RO	Reserved
0x030	PE2_CONFIG	RW	Processing Element 2 Static Config *Processing Element {0-3} Static Config (PE{0-3}_CFG) register* on page 11-180
0x034	PE2_RVBARADDR_LW	RW	Processing Element 2 Reset Vector Base Address Lower *Processing Element {0-3} Reset Vector Base Address Lower and Upper (PE{0-3}_RVBAR_LW) register* on page 11-181
0x038	PE2_RVBARADDR_UP	RO	Processing Element 2 Reset Vector Base Address Upper *Processing Element {0-3} Reset Vector Base Address Upper (PE{0-3}_RVBAR_UP) register* on page 11-182
0x03C	-	RO	Reserved
0x040	PE3_CONFIG	RW	Processing Element 3 Static Config *Processing Element {0-3} Static Config (PE{0-3}_CFG) register* on page 11-180
0x044	PE3_RVBARADDR_LW	RW	Processing Element 3 Reset Vector Base Address Lower *Processing Element {0-3} Reset Vector Base Address Lower and Upper (PE{0-3}_RVBAR_LW) register* on page 11-181
0x048	PE3_RVBARADDR_UP	RO	Processing Element 3 Reset Vector Base Address Upper *Processing Element {0-3} Reset Vector Base Address Upper (PE{0-3}_RVBAR_UP) register* on page 11-182
0x04C	-	RO	Reserved
0x050 – 0x1FC	-	RO	Reserved
0x200	HOST_RST_SYN	RO	Host Reset Syndrome *Host Reset Syndrome (HOST_RST_SYN) register* on page 11-182
0x204 – 0x2FC	-	RO	Reserved
0x300	HOST_CPU_BOOT_MSK	RW	Host CPU Boot Mask *Host CPU Boot Mask (HOST_CPU_BOOT_MSK) register* on page 11-183
0x304	HOST_CPU_CLUS_PWR_REQ	RW	Host CPU Cluster Power Request *Host CPU Cluster Power Request (HOST_CPU_CLUS_PWR_REQ) register* on page 11-183
0x308	HOST_CPU_WAKEUP	RW	Host CPU Wakeup *Host CPU Wakeup (HOST_CPU_WAKEUP) register* on page 11-184
0x30C	-	RO	Reserved
0x310	EXT_SYS0_RST_CTRL	RW	External System 0 Reset Control *External System {0-1} Reset Control (EXT_SYS{0-1}_RST_CTRL) register* on page 11-184
0x314	EXT_SYS0_RST_ST	RO	External System 0 Reset Status *External System {0-1} Reset Status (EXT_SYS{0-1}_RST_ST) register* on page 11-185
0x318	EXT_SYS1_RST_CTRL	RW	External System 1 Reset Control *External System {0-1} Reset Control (EXT_SYS{0-1}_RST_CTRL) register* on page 11-184

Table 11-10 Host Base System Control registers summary (continued)

Offset	Short name	Access	Name
0x31C	EXT_SYS1_RST_ST	RO	External System 1 Reset Status *External System {0-1} Reset Status (EXT_SYS{0-1}_RST_ST) register* on page 11-185
0x320 – 0x3FC	-	RO	Reserved
0x400	BSYS_PWR_REQ	RW	Base System Power Request *Base System Power Request (BSYS_PWR_REQ) register* on page 11-186
0x404	BSYS_PWR_ST	RO	Base System Power Status *Base System Power Status (BSYS_PWR_ST) register* on page 11-186
0x408 – 0x4FC	-	RO	Reserved
0x500	HOST_SYS_LCTRL_ST	RO	Host System Lock Control Status *Host System Lock Control Status (HOST_SYS_LCTRL_ST) register* on page 11-187
0x504	HOST_SYS_LCTRL_SET	WO	Host System Lock Control Set *Host System Lock Control Set (HOST_SYS_LCTRL_SET) register* on page 11-189
0x508	HOST_SYS_LCTRL_CLR	WO	Host System Lock Control Clear *Host System Lock Control Clear (HOST_SYS_LCTRL_CLR) register* on page 11-190
0x50C 0x– 0x7FC	-	RO	Reserved
0x800	HOSTCPUCLK_CTRL	RW	Host CPU Clock Control *Host CPU Clock Control (HOSTCPUCLK_CTRL) register* on page 11-192
0x804	HOSTCPUCLK_DIV0	RW	Host CPU Clock Divider 0 *Host CPU Clock Divider 0 (HOSTCPUCLK_DIV0) register* on page 11-193
0x808	HOSTCPUCLK_DIV1	RW	Host CPU Clock Divider 1 *Host CPU Clock Divider 1 (HOSTCPUCLK_DIV1) register* on page 11-193
0x80C	-	RO	Reserved
0x810	GICCLK_CTRL	RW	GIC Clock Control *GIC Clock Control (GICCLK_CTRL) register* on page 11-194
0x814	GICCLK_DIV0	RW	GIC Clock Divider 0 *GIC Clock Divider 0 (GICCLK_DIV0) register* on page 11-194
0x818 – 0x81C	-	RO	Reserved
0x820	ACLK_CTRL	RW	AXI Clock Control *AXI Clock Control (ACLK_CTRL) register* on page 11-195
0x824	ACLK_DIV0	RW	AXI Clock Divider 0 *AXI Clock Divider 0 (ACLK_DIV0) register* on page 11-196
0x828 – 0x82C	-	RO	Reserved
0x830	CTRLCLK_CTRL	RW	Control Clock Control *Control Clock Control (CTRLCLK_CTRL) register* on page 11-197
0x834	CTRLCLK_DIV0	RW	Control Clock Divider 0 *Control Clock Divider 0 (CTRLCLK_DIV0) register* on page 11-197

Table 11-10 Host Base System Control registers summary (continued)

Offset	Short name	Access	Name
0x838 – 0x83C	-	RO	Reserved
0x840	DBGCLK_CTRL	RW	Debug Clock Control *Debug Clock Control (DBGCLK_CTRL) register* on page 11-198
0x844	DBGCLK_DIV0	RW	Debug Clock Divider 0 *Debug Clock Divider 0 (DBGCLK_DIV0) register* on page 11-198
0x848 – 0x84C	-	RO	Reserved
0x850	HOSTUARTCLK_CTRL	RW	Host UART Clock Control *Host UART Clock Control (HOSTUARTCLK_CTRL) register* on page 11-199
0x854	HOSTUARTCLK_DIV0	RW	Host UART Clock Divider 0 *Host UART Clock Divider 0 (HOSTUARTCLK_DIV0) register* on page 11-200
0x858 – 0x85C	-	RO	Reserved
0x860	REFCLK_CTRL	RW	REFCLK Clock Control *REFCLK Clock Control register* on page 11-201
0x864 – 0x9FC	-	RO	Reserved
0xA00	CLKFORCE_ST	RO	Clock Force Status *Clock Force Status (CLKFORCE_ST) register* on page 11-201
0xA04	CLKFORCE_SET	WO	Clock Force Set *Clock Force Set (CLKFORCE_SET) register* on page 11-202
0xA08	CLKFORCE_CLR	WO	Clock Force Clear *Clock Force Clear (CLKFORCE_CLR) register* on page 11-203
0xA0C	-	RO	Reserved
0xA10	PLL_ST	RO	PLL Status *PLL Status (PLL_ST) register* on page 11-203
0xA14 – 0xAFC	-	RO	Reserved
0xB00	HOST_PPU_INT_ST	RO	Host PPU Interrupt Status *Host PPU Interrupt Status (HOST_PPU_INT_ST) register* on page 11-204
0xB04 – 0xFCC	-	RO	Reserved
0xFD0	PID4	RO	Peripheral ID4 *Peripheral ID 4 (PID4) register* on page 11-205
0xFD4	PID5	RO	Peripheral ID5 *Peripheral ID 5 (PID5) register* on page 11-205
0xFD8	PID6	RO	Peripheral ID6 *Peripheral ID 6 (PID6) register* on page 11-206
0xFDC	PID7	RO	Peripheral ID7 *Peripheral ID 7 (PID7) register* on page 11-206
0xFE0	PID0	RO	Peripheral ID0 *Peripheral ID 0 (PID0) register* on page 11-206
0xFE4	PID1	RO	Peripheral ID1 *Peripheral ID 1 (PID1) register* on page 11-206
0xFE8	PID2	RO	Peripheral ID2 *Peripheral ID 2 (PID2) register* on page 11-206
0xFEC	PID3	RO	Peripheral ID3 *Peripheral ID 3 (PID3) register* on page 11-207

Table 11-10 Host Base System Control registers summary (continued)

Offset	Short name	Access	Name
0xFF0	CID0	RO	Component ID0 *Component ID 0 (CID0) register* on page 11-207
0xFF4	CID1	RO	Component ID1 *Component ID 1 (CID1) register* on page 11-207
0xFF8	CID2	RO	Component ID2 *Component ID 2 (CID2) register* on page 11-207
0xFFC	CID3	RO	Component ID3 *Component ID 3 (CID3) register* on page 11-208

The Host Base System Control registers support 32-bit word aligned access. Any other bit size or unaligned access results in an error response and RAZ/WI.

The Host Base System Control registers have the following behaviors:

- Any read to a write-only register is treated as RAZ.
- Any write to a read-only register is treated as WI.

In both cases, no error is generated.

Cluster Static Config (CLUS_CFG) register

The following table gives a bit-level description of the Cluster Static Config register.

Table 11-11 CLUS_CFG register

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved.	RO	0x0000_000
[0]	CRYPTODISABLE	Disable the cryptographic extensions of the Host processor Cores. Possible field values are: 0b0 – Crypto is enabled. 0b1 – Crypto is disabled. Changes only take effect at power-on reset of the core.	RW	0b0

Note

A power-on reset of a core is when the CORE{0-3} power domain performs an OFF to ON transition.

When HOST_SYS_LCTRL.HOST_LOCK is set to 0b1, writing to this register does not update it and generates an error.

Processing Element {0-3} Static Config (PE{0-3}_CFG) register

The following table gives a bit-level description of the PE{0-3}_CFG register.

Table 11-12 PE{0-3}_CFG register

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RO	0x0000000
[3]	AA64nAA32	Reserved and treated as RAZ/WI	RW	0b0

Table 11-12 PE{0-3}_CFG register (continued)

Bits	Name	Description	Type	Reset
[2]	VINITHI	Locations of the exception vectors at reset. Sets the initial value of SCTLR.V. Possible field values are: 0: Exception vector start at `0x000000000` 1: Exception vector start at `0xFFFF00000` Changes in this field only take effect at a reset of the core.	RW	`0b0`
[1]	CFGTE	Enabling T32 exceptions. Sets the initial value of the SCTLR.TE. Possible field values are: 0: Exceptions taken in Arm(v7) or AArch32(v8) state 1: Exceptions taken in Thumb32 state Changes in this field only take effect at a reset of the core.	RW	`0b0`
[0]	CFGEND	Endianness configuration at reset. Sets the initial value of the SCTLR_EL3.EE and SCTR_S.EE bits. Possible field values are: 0: Little-endian 1: Big-endian Changes in this field only take effect at a reset of the core.	RW	`0b0`

Note

A power-on reset of a core is when the CORE{0-3} power domain performs an OFF to ON transition. A reset of the core is when the CORE{0-3} power domain performs any of the following transitions:

- OFF to ON
- OFF_EMU to ON
- WARM_RST to ON

When HOST_SYS_LCTRL_ST.HOST_LOCK is set to `0b1` a write to this register does not update the register and generates an error response.

Processing Element {0-3} Reset Vector Base Address Lower and Upper (PE{0-3}_RVBAR_LW) register

The following table gives a bit-level description of the Reset Vector Lower register.

Table 11-13 PE{0-3}_RVBAR_LW register

Bits	Name	Description	Type	Reset
[31:2]	RVBAR31_2	Reserved and treated as RAZ/WI	RW	`0x00000000`
[1:0]	-	Reserved	RO	`0b00`

Note

When HOST_SYS_LCTRL_ST.HOST_LOCK is set to `0b1` a write to this register does not update the register and generates an error response.

Processing Element {0-3} Reset Vector Base Address Upper (PE{0-3}_RVBAR_UP) register

The following table gives a bit-level description of the reset vector lower register.

Table 11-14 PE{0-3}_RVBAR_UP register

Bits	Name	Description	Type	Reset
[31:12]	-	Reserved	RO	`0x00000`
[11:0]	RVBAR43_32	Reset vector address bits [43:32]. Sets the initial value of RVBAR_EL3 register. As SSE-700 is a 32-bit system, this field is read-only and reads as all zeros.	RO	`0x000`

Note

When HOST_SYS_LCTRL_ST.HOST_LOCK, is set to `0b1` a write to this register does not update the register and generates an error response.

Host Reset Syndrome (HOST_RST_SYN) register

The following table gives a bit-level description of the Reset Syndrome register.

Table 11-15 HOST_RST_SYN register

Bits	Name	Description	Type	Reset
[31:4	-	Reserved.	RO	`0x0000000`
[3]	HOST	Indicates the last reset of the Host System was caused by the HOST_SYS_RST_CTRL.RST_REQ bit being set to `0b1`	RO	UNKNOWN
[2]	-	Reserved	RO	`0b0`
[1]	**nSRST**	Indicates that the last reset of the Host System was caused by either: • **nSRST** pin being asserted • **DP ROM CSYSRSTREQ** being asserted	RO	UNKNOWN
[0]	POR	Indicates that the last reset of the Host System was caused by one of the following: • **PORESETn** pin being asserted • **DP CDBGRSTREQ** being asserted • Secure Enclave Watchdog reset request • SoC Watchdog reset request • SOC_RST_CTRL.RST_REQ bit set to `0b1` • Secure Enclave software reset request	RO	UNKNOWN

Note

This register enables software to read the Reset Syndrome output of the Reset Controller. Therefore, the reset value of this register depends on the cause of the last reset of the Host System.

Host CPU Boot Mask (HOST_CPU_BOOT_MSK) register

The following table gives a bit-level description of the Boot Mask register.

Table 11-16 HOST_CPU_BOOT_MSK register

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved.	RO	0x0000000
[3:0]	BOOT_MSK	Configures which Host processor Cores automatically boot when the CLUSTOP domain exits OFF or MEM_RET. Each bit in this field corresponds to a Host processor Core, with bit 0 corresponding to Host processor Core 0 and bit 1 corresponding to Host processor Core 1. Bit values: 0: Host processor Core x does not automatically boot on CLUSTOP domain exit from OFF or MEM_RET. 1: Host processor Core x does automatically boot on CLUSTOP domain exit from OFF or MEM_RET. Changes in this field only take effect at the next CLUSTOP transitions from OFF or MEM_RET. The legal values depend on the number of Host processor cores used, see the table below. All other values are Reserved and generate an error if software attempts to write to the register.	RW	0x1

Note

When HOST_SYS_LCTRL_ST.HOST_LOCK is set to 1, a write to this register does not update the register and generates an error.

Table 11-17 Legal values of the BOOT MSK register

HOST_CPU_NUM_CORE	BOOT_MSK Legal Values
1	0x1
2	0x1 – 0x3
3	0x1 – 0x7
4	0x1 – 0xF

Host CPU Cluster Power Request (HOST_CPU_CLUS_PWR_REQ) register

The following table gives a bit-level description of the Cluster Power State Request register.

Table 11-18 HOST_CPU_CLUS_PWR_REQ register

Bits	Name	Description	Type	Reset
[31:2]	-	Reserved.	RO	0x00000000
[1]	MEM_RET_REQ	Possible field values are: 0: When entering a low power mode, the last level cache of the Host processor is not required to be retained 1: When entering a low power mode, the last level cache of the Host processor is required to be retained	RW	0b0
[0]	PWR_REQ	Possible field values are: 0: CLUSTOP can enter a low power mode, when all CORE{0-3} domains are OFF and the GIC is idle 1: CLUSTOP is required to be in a power mode greater than or equal to FUNC_RET even when all CORE{0-3} domains are OFF	RW	0b0

Host CPU Wakeup (HOST_CPU_WAKEUP) register

The following table gives a bit-level description of the Host CPU Wakeup register.

Table 11-19 HOST_CPU_WAKEUP register

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RO	0x0000000
[3]	CORE3_WAKEUP	Wakeup Core 3. This field is only implemented when HOST_CPU_NUM_CORES is 3 otherwise it is Reserved and treated as RAZ/WI.	RW	0b0
[2]	CORE2_WAKEUP	Wakeup Core 2. This field is only implemented when HOST_CPU_NUM_CORES > 2, otherwise it is Reserved and treated as RAZ/WI.	RW	0b0
[1]	CORE1_WAKEUP	Wakeup Core 1. This field is only implemented when HOST_CPU_NUM_CORES > 1, otherwise it is Reserved and treated as RAZ/WI.	RW	0b0
[0]	CORE0_WAKEUP	Wakeup Core 0	RW	0b0

Arm recommends, that software only writes to this register to wake a Host processor core for the first time after the CLUSTOP power domain has exited one of the following power modes: OFF, MEM_RET or WARM_RST. In all other cases, Arm recommends using interrupts through the Host processor GIC.

External System {0-1} Reset Control (EXT_SYS{0-1}_RST_CTRL) register

The following table gives a bit-level description of the Reset Control register.

Table 11-20 EXT_SYS{0-1}_RST_CTRL register

Bits	Name	Description	Type	Reset
[31:2]	-	Reserved	RO	`0x00000000`
[1]	RST_REQ	Reset request for External System. Possible field values are: 0: No reset requested 1: Reset requested	RW	`0b0`
[0]	CPUWAIT	CPU Wait control Possible field values are: 0: CPUWAIT signal of the External System is de-asserted 1: CPUWAIT signal of the External System is asserted When CPUWAIT becomes `0b0` any attempt to revert to `0b1` is ignored. This field only returns to `0b1` when any of the following occur: • External Power-on reset. • Internal Power-on reset. • Debug reset. • Host System reset. • Reset of the associated External System. This field becomes RO when associated EXT_SYS{0-1}_CPUWAIT_WEN is `0b0`. Any attempt to set this field to `0b0` by software is ignored.	RW	`0b1`

External System {0-1} Reset Status (EXT_SYS{0-1}_RST_ST) register

The following table gives a bit-level description of the Reset Status register.

Table 11-21 EXT_SYS{0-1}_RST_ST register

Bits	Name	Description	Type	Reset
[31:2]	-	Reserved	RO	0x000000
[2:1]	RST_ACK	Status of reset request Possible field values are: 0b00: No reset requested 0b01: Reset request unable to complete 0b10: Reset request complete 0b11: Reserved	RO	0b00
[0]	-	Reserved	RO	0b0

Base System Power Request (BSYS_PWR_REQ) register

The following table gives a bit-level description of the Base Sytem Power Request register.

Table 11-22 BSYS_PWR_REQ register

Bits	Name	Description	Type	Reset
[31:6]	-	Reserved	RO	0x0000000
[5:3]	SYSTOP_PWR_REQ	Selects SYSTOP power domain behavior when no activity in the domain. Possible field values are: 0b000: No request for logic or volatile memory to be powered 0b001: No request for logic to be powered, but volatile memory must be retained 0b01x: Request for logic to be powered, but volatile memory can be either powered or retained 0b1xx: Request for logic and volatile memory to be powered	RW	0b000
[2]	DBGTOP_PWR_REQ	Selects DBGTOP power domain behavior when no activity in the domain. Possible field values are: 0b0: No request for DBGTOP to be powered 0b1: Request for DBGTOP to be powered	RW	0b0
[1]	REFCLK_REQ	Request **REFCLK** Possible field values are: 0b0: No request for **REFCLK** to be supplied 0b1: Request for **REFCLK** to be supplied	RW	0b0
[0]	WAKEUP_EN	Host System wakeup enable. Possible field values are: 0b0: Wakeup for Host System is disabled 0b1: Wakeup for Host System is enabled	RW	0b0

Base System Power Status (BSYS_PWR_ST) register

The following table gives a bit-level description of the Bit level Base System Power Status register.

Table 11-23 BSYS_PWR_ST register

Bits	Name	Description	Type	Reset
[31:6]	-	Reserved	RO	0x0000000
[5:3]	SYSTOP_PWR_ST	SYSTOP power domain status. Possible field values are: 0b000: SYSTOP is in the OFF or WARM_RST power mode 0b001: SYSTOP is in the MEM_RET power mode 0b010: SYSTOP is in the FUNC_RET power mode 0b100: SYSTOP is in the ON power mode All other values are Reserved.	RO	UNKNOWN
[2]	DBGTOP_PWR_ST	DBGTOP power domain status. Possible field values are: 0b0: DBGTOP is in the OFF or WARM_RST power mode 0b1: DBGTOP is in the ON-power mode	RO	UNKNOWN
[1]	-	Reserved	RO	0b0
[0]	-	Reserved	RO	0b0

Note

The values in this register are driven from the PPUHWSTAT outputs of the respective PPU, and are only valid if the respective PPU is not making a transition.

Host System Lock Control Status (HOST_SYS_LCTRL_ST) register

The following table gives a bit-level description of the Lock Control register.

Table 11-24 HOST_SYS_LCTRL_ST register

Bits	Name	Description	Type	Reset
[31]	LOCK_CLR_DIS	Controls the behavior of the HOST_SYS_LCTRL_CLR register 0b0: Writes to the HOST_SYS_LCTRL_CLR register take effect 0b1: Writes to the HOST_SYS_LCTRL_CLR register are ignored	RO	0b0
[30:8]	-	Reserved	RO	0x000000
[7]	HOST_LOCK	Controls whether registers in the Host Base System Control, which are lockable, are locked or unlocked. 0b0: Registers are unlocked 0b1: Registers are locked	RO	0b0

Table 11-24 HOST_SYS_LCTRL_ST register (continued)

Bits	Name	Description	Type	Reset
[6]	HOST_GIC_LOCK	Drives the **CFGSDISABLE** signal of the GICCFG interface. `0b0`: Signal is de-asserted. `0b1`: Signal is asserted. This field is set to `0b0` when CLUSTOP power domain enters on of the following: • OFF • MEM_RET • WARM_RST	RO	`0b0`
[5]	HOST_CPU3_LOCK	Drives the **CP15SDISABLE[3]** signal of the HOSTCPUCFG interface. `0b0`: Signal is de-asserted. `0b1`: Signal is asserted. This field is set to `0b0` when: • CORE3 power domain enters OFF, OFF_EMU or WARM_RST. • CLUSTOP power domain enters OFF, MEM_RET or WARM_RST. This field is Reserved and treated as RAZ/WI when HOST_CPU_NUM_CORE < 4.	RO	`0b0`
[4]	HOST_CPU2_LOCK	Drives the **CP15SDISABLE[2]** signal of the HOSTCPUCFG interface. `0b0`: Signal is de-asserted. `0b1`: Signal is asserted. This field is set to `0b0` when: • CORE2 power domain enters OFF, OFF_EMU, or WARM_RST • CLUSTOP power domain enters OFF, MEM_RET, or WARM_RST This field is Reserved and treated as RAZ/WI when HOST_CPU_NUM_CORE < 3.	RO	`0b0`
[3]	HOST_CPU1_LOCK	Drives the **CP15SDISABLE[1]** signal of the HOSTCPUCFG interface. `0b0`: Signal is de-asserted. `0b1`: Signal is asserted. This field is set to `0b0` when: • CORE1 power domain enters OFF, OFF_EMU, or WARM_RST • CLUSTOP power domain enters OFF, MEM_RET, or WARM_RST This field is Reserved and treated as RAZ/WI when HOST_CPU_NUM_CORE < 2.	RO	`0b0`
[2]	HOST_CPU0_LOCK	Drives the **CP15SDISABLE[0]** signal of the HOSTCPUCFG interface. `0b0`: Signal is de-asserted. `0b1`: Signal is asserted. This field is set to `0b0` when: • CORE0 power domain enters OFF, OFF_EMU, or WARM_RST • CLUSTOP power domain enters OFF, MEM_RET, or WARM_RST	RO	`0b0`

Table 11-24 HOST_SYS_LCTRL_ST register (continued)

Bits	Name	Description	Type	Reset
[1]	INT_RTR_LOCK	Controls the Interrupt Router Lockdown interface. `0b0`: Interrupt Router Lockdown interface is de-asserted `0b1`: Interrupt Router Lockdown interface is asserted	RO	`0b0`
[0]	HOST_FW_LOCK	Controls the Host Firewall Lockdown interface. `0b0`: Host Firewall Lockdown interface is de-asserted `0b1`: Host Firewall Lockdown interface is asserted	RO	`0b0`

Note

If there is a simultaneous set and clear event, where the clear event can be either a write to the HOST_SYS_LCTRL_CLR register or a condition described in the above table, then the clear event always takes precedence. For information on how software should use the HOST_SYS_LCTRL_{ST/SET/CLR} registers see *Chapter 12 Software sequences* on page 12-238

Host System Lock Control Set (HOST_SYS_LCTRL_SET) register

The following table gives a bit-level description of the Lock Control Set register.

Table 11-25 HOST_SYS_LCTRL_SET register

Bits	Name	Description	Type	Reset
[31]	LOCK_CLR_DIS	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.LOCK_CLR_DIS field to 1. Writing 0 to this field has no effect on the value of HOST_SYS_LCTRL_ST.LOCK_CLR_DIS field. This field always reads as 0.	WO	`0b0`
[30:8]	-	Reserved	RO	`0x000000`
[7]	HOST_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_LOCK field to 1. Writing 0 to this field has no effect on the value of HOST_SYS_LCTRL_ST.HOST_LOCK field. This field always reads as 0.	WO	`0b0`
[6]	HOST_GIC_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_GIC_LOCK field to 1. Writing 0 to this field has no effect on the value of HOST_SYS_LCTRL_ST.HOST_GIC_LOCK field. This field always reads as 0.	WO	`0b0`

Table 11-25 HOST_SYS_LCTRL_SET register (continued)

Bits	Name	Description	Type	Reset
[5]	HOST_CPU3_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_CPU3_LOCK field to 1. Writing 0 to this field has no effect on the value of HOST_SYS_LCTRL_ST.HOST_CPU3_LOCK field. This field always reads as 0.	WO	0b0
[4]	HOST_CPU2_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_CPU2_LOCK field to 1. Writing 0 to this field has no effect on the value of HOST_SYS_LCTRL_ST.HOST_CPU2_LOCK field. This field always reads as 0.	WO	0b0
[3]	HOST_CPU1_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_CPU1_LOCK field to 1. Writing 0 to this field has no effect on the value of HOST_SYS_LCTRL_ST.HOST_CPU1_LOCK field. This field always reads as 0.	WO	0b0
[2]	HOST_CPU0_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_CPU0_LOCK field to 1. Writing 0 to this field has no effect on the value of HOST_SYS_LCTRL_ST.HOST_CPU0_LOCK field. This field always reads as 0.	WO	0b0
[1]	INT_RTR_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.INT_RTR_LOCK field to 1. Writing 0 to this field has no effect on the value of HOST_SYS_LCTRL_ST.INT_RTR_LOCK field. This field always reads as 0.	WO	0b0
[0]	HOST_FW_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_FW_LOCK field to 1. Writing 0 to this field has no effect on the value of HOST_SYS_LCTRL_ST.HOST_FW_LOCK field. This field always reads as 0.	WO	0b0

Host System Lock Control Clear (HOST_SYS_LCTRL_CLR) register

The following table gives a bit-level description of the Lock Control Clear register.

Table 11-26 HOST_SYS_LCTRL_CLR register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x000000
[7]	HOST_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_LOCK field to 0. Writing 0 to this field has no effect on the value of the HOST_SYS_LCTRL_ST.HOST_LOCK field. This field always reads as 0.	WO	0b0
[6]	HOST_GIC_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_GIC_LOCK field to 0. Writing 0 to this field has no effect on the value of the HOST_SYS_LCTRL_ST.HOST_GIC_LOCK field. This field always reads as 0.	WO	0b0
[5]	HOST_CPU3_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_CPU3_LOCK field to 0. Writing 0 to this field has no effect on the value of the HOST_SYS_LCTRL_ST.HOST_CPU3_LOCK field. This field always reads as 0.	WO	0b0
[4]	HOST_CPU2_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_CPU2_LOCK field to 0. Writing 0 to this field has no effect on the value of the HOST_SYS_LCTRL_ST.HOST_CPU2_LOCK field. This field always reads as 0.	WO	0b0
[3]	HOST_CPU1_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_CPU1_LOCK field to 0. Writing 0 to this field has no effect on the value of the HOST_SYS_LCTRL_ST.HOST_CPU1_LOCK field. This field always reads as 0.	WO	0b0
[2]	HOST_CPU0_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_CPU0_LOCK field to 0. Writing 0 to this field has no effect on the value of the HOST_SYS_LCTRL_ST.HOST_CPU0_LOCK field. This field always reads as 0.	WO	0b0

Table 11-26 HOST_SYS_LCTRL_CLR register (continued)

Bits	Name	Description	Type	Reset
[1]	INT_RTR_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.INT_RTR_LOCK field to 0. Writing 0 to this field has no effect on the value of the HOST_SYS_LCTRL_ST.INT_RTR_LOCK field. This field always reads as 0.	WO	0b0
[0]	HOST_FW_LOCK	Writing 1 to this field sets the HOST_SYS_LCTRL_ST.HOST_FW_LOCK field to 0. Writing 0b0 to this field has no effect on the value of the HOST_SYS_LCTRL_ST.HOST_FW_LOCK field. This field always reads as 0.	WO	0b0

Note

Writes to the HOST_SYS_LCTRL_CLR register are ignored when the HOST_SYS_LCTRL_ST.LOCK_CLR_DIS field is set to 1.

Host CPU Clock Control (HOSTCPUCLK_CTRL) register

The following table gives a bit-level description of the Host CPU Clock Control register.

Table 11-27 HOSTCPUCLK_CTRL register

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RO	0x0000
[15:8]	CLKSELECT_CUR	Currently selected clock source for **HOSTCPUCLK**. Possible field values are: 0x00: Clock gate 0x01: **REFCLK** 0x02: **SYSPLL** 0x04: **CPUPLL** All other values are Reserved.	RO	UNKNOWN
[7:0]	CLKSELECT	Select the clock source for **HOSTCPUCLK**. Possible field values are: 0x00: Clock gate 0x01: **REFCLK** 0x02: **SYSPLL** 0x04: **CPUPLL** All other values are Reserved. Selecting a value which is Reserved can cause a deadlock.	RW	0x01

Host CPU Clock Divider 0 (HOSTCPUCLK_DIV0) register

The following table gives a bit-level description of the Host CPU Clock Divider 0 register.

Table 11-28 HOSTCPUCLK_DIV0 register

Bits	Name	Description	Type	Reset
[31:21]	-	Reserved	RO	0x000
[20:16]	CLKDIV_CUR	Current value of integer divider applied to **SYSPLL**. Possible field values are: 0x00: Divide by 1 0x01: Divide by 2 … 0x0F: Divide by 32	RO	UNKNOWN
[15:5]	-	Reserved	RO	0x000
[4:0]	CLKDIV	Select the value of the integer divider applied to **SYSPLL**. Possible field values are: 0x00: Divide by 1 0x01: Divide by 2 … 0x1F: Divide by 32	RW	0x00

Host CPU Clock Divider 1 (HOSTCPUCLK_DIV1) register

The following table gives a bit-level description of the Host CPU Clock Divider 1 register.

Table 11-29 HOSTCPUCLK_DIV1 register

Bits	Name	Description	Type	Reset
[31:21]	-	Reserved	RO	0x000
[20:16]	CLKDIV_CUR	Current value of integer divider applied to **CPUPLL**. Possible field values are: 0x00: Divide by 1 0x01: Divide by 2 … 0x0F: Divide by 32	RO	UNKNOWN

Table 11-29 HOSTCPUCLK_DIV1 register (continued)

Bits	Name	Description	Type	Reset
[15:5]	-	Reserved	RO	0x000
[4:0]	CLKDIV	Select the value of the integer divider applied to **CPUPLL**. Possible field values are: 0x00: Divide by 1 0x01: Divide by 2 … 0x1F: Divide by 32	RW	0x00

GIC Clock Control (GICCLK_CTRL) register

The following table gives a bit-level description of the GIC Clock Control register.

Table 11-30 GICCLK_CTRL register

Bits	Name	Description	Type	Reset
[31:24]	ENTRY_DELAY	Reserved and treated as RAZ/WI	RW	0x00
[23:16]	-	Reserved	RO	0x00
[15:8]	CLKSELECT_CUR	Currently selected clock source for **GICCLK**. Possible field values are: 0x00: Clock gate 0x01: **REFCLK** 0x02: **SYSPLL** All other values are Reserved.	RO	UNKNOWN
[7:0]	CLKSELECT	Select the clock source for **GICCLK**. Possible field values are: 0x000: Clock gate 0x001: **REFCLK** 0x002: **SYSPLL** All other values are Reserved. Selecting a value which is Reserved can cause a deadlock.	RW	0x01

GIC Clock Divider 0 (GICCLK_DIV0) register

The following table gives a bit-level description of the GIC Clock Divider 0 register.

Table 11-31 GICCLK_DIV0 register

Bits	Name	Description	Type	Reset
[31:21]	-	Reserved.	RO	`0x000`
[20:16]	CLKDIV_CUR	Current value of integer divider applied to **SYSPLL**. Possible field values are: `0x00`: Divide by 1 `0x01`: Divide by 2 … `0x1F`: Divide by 32	RO	UNKNOWN
[15:5]	-	Reserved	RO	`0x000`
[4:0]	CLKDIV	Select the value of the integer divider applied to **SYSPLL**. Possible field values are: `0x00`: Divide by 1 `0x01`: Divide by 2 … `0x1F`: Divide by 32	RW	`0x00`

AXI Clock Control (ACLK_CTRL) register

The following table gives a bit-level description of the AXI Clock Control register.

Table 11-32 ACLK_CTRL register

Bits	Name	Description	Type	Reset
[31:24]	ENTRY_DELAY	Configure number of idle clock cycles before clock is gated	RW	`0x00`
[23:16]	-	Reserved	RO	`0x00`

Table 11-32 ACLK_CTRL register (continued)

Bits	Name	Description	Type	Reset
[15:8]	CLKSELECT_CUR	Currently selected clock source for **ACLK**. Possible field values are: `0x00`: Clock gate `0x01`: **REFCLK** `0x02`: **SYSPLL** All other values are Reserved.	RO	UNKNOWN
[7:0]	CLKSELECT	Select the clock source for **ACLK**. Possible field values are: `0x00`: Clock gate `0x01`: **REFCLK** `0x02`: **SYSPLL** All other values are Reserved. Selecting a value which is Reserved can cause a deadlock.	RW	`0x01`

AXI Clock Divider 0 (ACLK_DIV0) register

The following table gives a bit-level description of the AXI Clock Divider 0 register description.

Table 11-33 ACLK_DIV0 register

Bits	Name	Description	Type	Reset
[31:21]	-	Reserved	RO	`0x000`
[20:16]	CLKDIV_CUR	Current value of integer divider applied to **SYSPLL**. Possible field values are: `0x00`: Divide by 1. `0x01`: Divide by 2. … `0x1F`: Divide by 32.	RO	UNKNOWN
[15:5]	-	Reserved	RO	`0x000`
[4:0]	CLKDIV	Select the value of the integer divider applied to **SYSPLL**. Possible field values are: `0x00`: Divide by 1 `0x01`: Divide by 2 … `0x1F`: Divide by 32	RW	`0x00`

Control Clock Control (CTRLCLK_CTRL) register

The following table gives a bit-level description of the Control Clock Control register.

Table 11-34 CTRLCLK_CTRL register

Bits	Name	Description	Type	Reset
[31:24]	ENTRY_DELAY	Configure number of idle clock cycles before clock is gated	RW	0x00
[23:16]	-	Reserved	RO	0x00
[15:8]	CLKSELECT_CUR	Currently selected clock source for **CTRLCLK**. Possible field values are: 0x00: Clock gate 0x01: **REFCLK** 0x02: **SYSPLL** All other values are Reserved.	RO	UNKNOWN
[7:0]	CLKSELECT	Select the clock source for **CTRLCLK**. Possible field values are: 0x00: Clock gate 0x01: **REFCLK** 0x02: **SYSPLL** All other values are Reserved. Selecting a value which is Reserved can cause a deadlock.	RW	0x01

Control Clock Divider 0 (CTRLCLK_DIV0) register

The following table gives a bit-level description of the Control Clock Divider 0 register.

Table 11-35 CTRLCLK_DIV0 register

Bits	Name	Description	Type	Reset
[31:21]	-	Reserved	RO	Reserved
[20:16]	CLKDIV_CUR	Current value of integer divider applied to **SYSPLL**. Possible field values are: 0x00: Divide by 1 0x01: Divide by 2 … 0x1F: Divide by 32	RO	UNKNOWN

Table 11-35 CTRLCLK_DIV0 register (continued)

Bits	Name	Description	Type	Reset
[15:5]	-	Reserved	RO	0x000
[4:0]	CLKDIV	Select the value of the integer divider applied to **SYSPLL**. Possible field values are: 0x00: Divide by 1 0x01: Divide by 2 … 0x1F: Divide by 32	RW	0x00

Debug Clock Control (DBGCLK_CTRL) register

The following table gives a bit-level description of the Debug Clock Control register.

Table 11-36 DBGCLK_CTRL register

Bits	Name	Description	Type	Reset
[31:24]	ENTRY_DELAY	Configure number of idle clock cycles before clock is gated	RW	0x00
[23:16]	-	Reserved	RO	0x00
[15:8]	CLKSELECT_CUR	Currently selected clock source for **DBGCLK**. Possible field values are: 0x00: Clock gate 0x01: **REFCLK** 0x02: **SYSPLL** All other values are Reserved.	RO	UNKNOWN
[7:0]	CLKSELECT	Select the clock source for **DBGCLK**. Possible field values are: 0x00: Clock gate 0x01: **REFCLK** 0x02: **SYSPLL** All other values are Reserved. Selecting a value which is Reserved can cause a deadlock.	RW	0x01

Debug Clock Divider 0 (DBGCLK_DIV0) register

The following table gives a bit-level description of the Debug Clock Divider 0 register.

Table 11-37 DBGCLK_DIV0 register

Bits	Name	Description	Type	Reset
[31:21]	-	Reserved	RO	`0x000`
[20:16]	CLKDIV_CUR	Current value of integer divider applied to **SYSPLL**. Possible field values are: `0x00`: Divide by 1 `0x01`: Divide by 2 … `0x1F`: Divide by 32	RO	UNKNOWN
[15:5]	-	Reserved	RO	`0x000`
[4:0]	CLKDIV	Select the value of the integer divider applied to **SYSPLL**. Possible field values are: `0x00`: Divide by 1. `0x01`: Divide by 2. … `0x1F`: Divide by 32.	RW	`0x00`

Host UART Clock Control (HOSTUARTCLK_CTRL) register

The following table gives a bit-level description of the Host UART Clock Control register.

Table 11-38 HOSTUARTCLK_CTRL register

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RO	0x0000
[15:8]	CLKSELECT_CUR	Currently selected clock source for **HOSTUARTCLK**. Possible field values are: 0x00: Clock gate 0x01: **REFCLK** 0x02: **UARTCLK** 0x04: **S32KCLK** All other values are Reserved.	RO	UNKNOWN
[7:0]	CLKSELECT	Select the clock source for **HOSTUARTCLK**. Possible field values are: 0x00: Clock gate 0x01: **REFCLK** 0x02: **UARTCLK** 0x04: **S32KCLK** All other values are Reserved. Selecting a value which is Reserved can cause a deadlock.	RW	0x01

Host UART Clock Divider 0 (HOSTUARTCLK_DIV0) register

The following table gives a bit-level description of the Host UART Clock Divider 0 register.

Table 11-39 HOSTUARTCLK_DIV0 register

Bits	Name	Description	Type	Reset
[31:21]	-	Reserved	RO	0x000
[20:16]	CLKDIV_CUR	Current value of integer divider applied to **UARTCLK**. Possible field values are: 0x00: Divide by 1 0x01: Divide by 2 … 0x1F: Divide by 32	RO	UNKNOWN

Table 11-39 HOSTUARTCLK_DIV0 register (continued)

Bits	Name	Description	Type	Reset
[15:5]	-	Reserved	RO	0x000
[4:0]	CLKDIV	Select the value of the integer divider applied to **UARTCLK**. Possible field values are: 0x00: Divide by 1 0x01: Divide by 2 … 0x1F: Divide by 32	RW	0x00

REFCLK Clock Control register

The following table gives a bit-level description of the REFCLK Clock Control register.

Table 11-40 REFLCK register

Bits	Name	Description	Type	Reset
[31:24]	ENTRY_DELAY	Configure number of idle clock cycles before clock is gated	RW	0x00
[23:16]	-	Reserved	RO	0x00
[15:8]	CLKSELECT_CUR	Currently selected clock source for **DBGCLK**. 0x01: **REFCLK** All other values are Reserved.	RO	UNKNOWN
[7:0]	CLKSELECT	Select the clock source for **DBGCLK:** 0x01: **REFCLK** All other values are Reserved.	RO	0x01

Clock Force Status (CLKFORCE_ST) register

The following table gives a bit-level description of the Clock Force Status register.

Table 11-41 CLKFORCE_ST register

Bits	Name	Description	Type	Reset
[31:7]	-	Reserved	RO	0x0000000
[6]	REFCLK_FORCE_ST	Status of **REFCLK** clock force. 0b0: High-level clock gating is enabled. 0b1: High-level clock gating is disabled.	RO	0b0
[5]	-	Reserved	RO	0b0

Table 11-41 CLKFORCE_ST register (continued)

Bits	Name	Description	Type	Reset
[4]	DBGCLK_FORCE_ST	Status of **DBGCLK** clock force. Possible field values are: `0b0`: High-level clock gating is enabled. `0b1` – High-level clock gating is disabled.	RO	`0b0`
[3]	CTRLCLK_FORCE_ST	Status of **CTRLCLK** clock force. Possible field values are: `0b0`: High-level clock gating is enabled. `0b1`: High-level clock gating is disabled.	RO	`0b0`
[2]	ACLK_FORCE_ST	Status of **ACLK** clock force. Possible field values are: `0b0`: High-level clock gating is enabled. `0b1`: High-level clock gating is disabled.	RO	`0b0`
[1]	GICCLK_FORCE_ST	Reserved and treated as RAZ/WI	RO	`0b0`
[0]	-	Reserved	RO	`0b0`

Clock Force Set (CLKFORCE_SET) register

The following table gives a bit-level description of the Clock Force Set register.

Table 11-42 CLKFORCE_SET register

Bits	Name	Description	Type	Reset
[31:7]	-	Reserved	RO	`0x0000000`
[6]	REFCLK_FORCE_SET	Set REFCLKCLK_FORCE_ST in CLKFORCE_ST register. Writing a value of `0b0` has no effect. This field always reads as `0b0`.	WO	`0b0`
[5]	-	Reserved	RO	`0b0`
[4]	DBGCLK_FORCE_SET	Set DBGCLK_FORCE_ST in CLKFORCE_ST register. Writing a value of `0b0` has no effect. This field always reads as`0b 0`.	WO	`0b0`
[3]	CTRLCLK_FORCE_SET	Set CTRLCLK_FORCE_ST in CLKFORCE_ST register. Writing a value of `0b0` has no effect. This field always reads as `0b0`.	WO	`0b0`

Table 11-42 CLKFORCE_SET register (continued)

Bits	Name	Description	Type	Reset
[2]	ACLK_FORCE_SET	Set ACLK_FORCE_ST in CLKFORCE_ST register. Writing a value of `0b0` has no effect. This field always reads as `0b0`.	WO	`0b0`
[1]	GICCLK_FORCE_SET	Reserved and treated as RAZ/WI	WO	`0b0`
[0]	-	Reserved	WO	`0b0`

Clock Force Clear (CLKFORCE_CLR) register

The following table gives a bit-level description of the Clock Force Clear register description.

Table 11-43 CLKFORCE_CLR register

Bits	Name	Description	Type	Reset
[31:7]	-	Reserved	RO	`0x0000000`
[6]	REFCLK_FORCE_CLR	Clear REFCLK_FORCE_ST in CLKFORCE_ST register. Writing a value of `0b0` has no effect. This field always reads as `0b0`.	WO	`0b0`
[5]	-	Reserved	RO	`0b0`
[4]	DBGCLK_FORCE_CLR	Clear DBGCLK_FORCE_ST in CLKFORCE_ST register. Writing a value of `0b0` has no effect. This field always reads as `0b0`.	WO	`0b0`
[3]	CTRLCLK_FORCE_CLR	Clear CTRLCLK_FORCE_ST in CLKFORCE_ST register. Writing a value of `0b0` has no effect. This field always reads as `0b0`.	WO	`0b0`
[2]	ACLK_FORCE_CLR	Clear ACLK_FORCE_ST in CLKFORCE_ST register. Writing a value of `0b0` has no effect. This field always reads as `0b0`.	WO	`0b0`
[1]	GICCLK_FORCE_CLR	Reserved and treated as RAZ/WI	WO	`0b0`
[0]	-	Reserved	RO	`0b0`

PLL Status (PLL_ST) register

The following table gives a bit-level description of the Phase Locked Loop Status register.

Table 11-44 PLL_ST register

Bits	Name	Description	Type	Reset
[31:2]	-	Reserved	RO	0x0000000
[1]	CPUPLLLOCK_ST	Status of the **CPUPLLLOCK** input. Possible field values are: 0: **PLL** is not locked. 1: **PLL** is locked.	RO	UNKNOWN
[0]	SYSPLLLOCK_ST	Status of the **SYSPLLLOCK** input. Possible field values are: 0: **PLL** is not locked. 1: **PLL** is locked.	RO	UNKNOWN

Host PPU Interrupt Status (HOST_PPU_INT_ST) register

The following table gives a bit-level description of the PPU Interrupt Status register.

Table 11-45 HOST_PPU_INT_ST register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved.	RO	0x000000
[7]	CORE3_INT_ST	Status of CORE3 PPU Interrupt. Possible field values are: 0: Interrupt is de-asserted 1: Interrupt is asserted This field is Reserved and treated as RAZ/WI when HOST_CPU_NUM_CORE < 4.	RO	UNKNOWN
[6]	CORE2_INT_ST	Status of CORE2 PPU Interrupt. Possible field values are: 0: Interrupt is de-asserted 1: Interrupt is asserted This field is Reserved and treated as RAZ/WI when HOST_CPU_NUM_CORE < 3.	RO	UNKNOWN
[5]	CORE1_INT_ST	Status of CORE1 PPU Interrupt. Possible field values are: 0: Interrupt is de-asserted 1: Interrupt is asserted This field is Reserved and treated as RAZ/WI when HOST_CPU_NUM_CORE < 2.	RO	UNKNOWN

Table 11-45 HOST_PPU_INT_ST register (continued)

Bits	Name	Description	Type	Reset
[4]	CORE0_INT_ST	Status of CORE0 PPU Interrupt. Possible field values are: 0: Interrupt is de-asserted 1: Interrupt is asserted	RO	UNKNOWN
[3]	CLUSTOP_INT_ST	Status of CLUSTOP PPU Interrupt. Possible field values are: 0: Interrupt is de-asserted 1: Interrupt is asserted	RO	UNKNOWN
[2]	SYSTOP_INT_ST	Status of SYSTOP PPU Interrupt. Possible field values are: 0: Interrupt is de-asserted 1: Interrupt is asserted	RO	UNKNOWN
[1]	DBGTOP_INT_ST	Status of the DBGTOP PPU Interrupt. Possible field values are: 0: Interrupt is de-asserted 1: Interrupt is asserted	RO	UNKNOWN
[0]	FW_INT_ST	Status of the Firewall PPU Interrupt. Possible field values are: 0: Interrupt is de-asserted 1: Interrupt is asserted	RO	UNKNOWN

When any bit in this register is 1, the PPU Combined interrupt will be asserted.

Peripheral ID 4 (PID4) register

The following table gives a bit-level description of the Peripheral ID 4 register.

Table 11-46 PID4 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	`0x000000`
[7:4]	Size	Number of 4KB blocks occupied by the System ID block. This field is deprecated.	RO	`0x0`
[3:0]	DES_2	JEP Continuation	RO	`0x4`

Peripheral ID 5 (PID5) register

The following table gives a bit-level description of the Peripheral ID 5 register.

Table 11-47 PID5 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x00000000

Peripheral ID 6 (PID6) register

The following table gives a bit-level description of the Peripheral ID 6 register.

Table 11-48 PID6 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x00000000

Peripheral ID 7 (PID7) register

The following table gives a bit-level description of the Peripheral ID 7 register.

Table 11-49 PID7 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x00000000

Peripheral ID 0 (PID0) register

The following table gives a bit-level description of the Peripheral ID 0 register.

Table 11-50 PID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x000000
[7:0]	PART_0	Bits [7:0] of part ID	RO	0x78

Peripheral ID 1 (PID1) register

The following table gives a bit-level description of the Peripheral ID 1 register.

Table 11-51 PID1 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x000000
[7:4]	DES_0	Bits [3:0] of JEP 106 Identity	RO	0xB
[3:0]	PART_1	Bits [11:8] of part ID	RO	0x0

Peripheral ID 2 (PID2) register

The following table gives a bit-level description of the Peripheral ID 2 register.

Table 11-52 PID2 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x000000
[7:4]	REVISION	Major revision of the System ID block	RO	0x0b0
[3]	JEDEC	Indicates the use of JEDEC JEP106 identification scheme	RO	1
[2:0]	DES_1	Bits [6:4] of JEP 106 Identity	RO	0b011

Peripheral ID 3 (PID3) register

The following table gives a bit-level description of the Peripheral ID 3 register.

Table 11-53 PID3 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x000000
[7:4]	REVAND	Minor revision of the System ID block	RO	0x0
[3:0]	CMOD	Customer modification field	RO	0x0

Component ID 0 (CID0) register

The following table gives a bit-level description of the Component ID 0 register.

Table 11-54 CID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x000000
[7:0]	PRMBL_0	Preamble 0	RO	0x0D

Component ID 1 (CID1) register

The following table gives a bit-level description of the Component ID 1 register.

Table 11-55 CID1 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x000000
[7:4]	CLASS	Class of the component	RO	0xF
[3:0]	PRMBL_1	Preamble 1	RO	0x0

Component ID 2 (CID2) register

The following table gives a bit-level description of the Component ID 2 register.

Table 11-56 CID2 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x000000
[7:0]	PRMBL_2	Preamble 2	RO	0x05

Component ID 3 (CID3) register

The following table gives a bit-level description of the Component ID 3 register.

Table 11-57 CID3 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x000000
[7:0]	PRMBL_3	Preamble 3	RO	0xB1

11.3.2 Secure Enclave System and Base System Control registers

Secure Enclave System Control and Base System Control registers reside in the Secure Enclave memory Map.

For detailed information see the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual* .

11.3.3 Interrupt Router register summary

This section describes the register set of the Interrupt Router.

The Interrupt Router is in the AONTOP power domain. It is accessed at offset 0x1A50_0000 in the Host System memory map.

Table 11-58 Interrupt Router register summary

Offset	Short Name	Access	Name
0x000	INT_RTR_CTRL	RW	Interrupt Router Control *Interrupt Router Control (INT_RTR_CTRL) register* on page 11-209
0x004 – 0x00C	-	RO	Reserved
0x010	LD_CTRL	RW	Lockdown Control *Lockdown Control (LD_CTRL) register* on page 11-210
0x014 – 0x0FC	-	RO	Reserved
0x100	SHD_INT_INFO	RO	Shared Interrupt Information *Shared Interrupt Information (SHD_INT_INFO) register* on page 11-210
0x104	SHD_INT_CFG	RW	Shared Interrupt Configuration *Shared Interrupt Configuration (SHD_INT_CFG) register* on page 11-210
0x108	SHD_INT_LCTRL	RW	Shared Interrupt Lock Control *Shared Interrupt Lock Control (SHD_INT_LCTRL) register* on page 11-211
0x10C	SHD_INT_SEL	RW	Shared Interrupt Select *Shared Interrupt Select (SHD_INT_SEL) register* on page 11-211
0x110 – 0xE8C	-	RO	Reserved
0xE90	INT_RTR_TMP_ST	RW	Interrupt Router Tamper Status *Interrupt Router Tamper Status (INT_RTR_TMP_ST) register* on page 11-211

Table 11-58 Interrupt Router register summary (continued)

Offset	Short Name	Access	Name
0xE94 – 0xF9C	-	RO	Reserved
0xFA0	INT_RTR_CAP	RO	Interrupt Router Capability *Interrupt Router Capability (INT_RTR_CAP) register* on page 11-212
0xFA4 – 0xFAC	-	RO	Reserved
0xFB0	INT_RTR_CFG	RO	Interrupt Router Configuration *Interrupt Router Configuration (INT_RTR_CFG) register* on page 11-212
0xFB4 – 0xFCC	-	RO	Reserved
0xFD0	PID4	RO	Peripheral ID4 *Peripheral ID 4 (PID4) register* on page 11-213
0xFD4	PID5	RO	Peripheral ID5 *Peripheral ID 5 (PID5) register* on page 11-213
0xFD8	PID6	RO	Peripheral ID6 *Peripheral ID 6 (PID6) register* on page 11-213
0xFDC	PID7	RO	Peripheral ID7 *Peripheral ID 7 (PID7) register* on page 11-213
0xFE0	PID0	RO	Peripheral ID0 *Peripheral ID 0 (PID0) register* on page 11-214
0xFE4	PID1	RO	Peripheral ID1 *Peripheral ID 1 (PID1) register* on page 11-214
0xFE8	PID2	RO	Peripheral ID2 *Peripheral ID 2 (PID2) register* on page 11-214
0xFEC	PID3	RO	Peripheral ID3 *Peripheral ID 3 (PID3) register* on page 11-214
0xFF0	CID0	RO	Component ID0 *Component ID 0 (CID0) register* on page 11-215
0xFF4	CID1	RO	Component ID1 *Component ID 1 (CID1) register* on page 11-215
0xFF8	CID2	RO	Component ID2 *Component ID 2 (CID2) register* on page 11-215
0xFFC	CID3	RO	Component ID3 *Component ID 3 (CID3) register* on page 11-215

Note

These registers only support 32-bit word aligned access. Any other attempts at access generate an error and are treated as RAZ/WI. For more information on access to the registers of the Interrupt Router see *Appendix B Interrupt Router* on page Appx-B-274.

Interrupt Router Control (INT_RTR_CTRL) register

The following table gives a bit-level description of the Interrupt Router Control register.

Table 11-59 INT_RTR_CTRL register

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved	RO	0x0000_0000
[0]	ERR	Configures the response for Configuration Accesses which generate a Configuration Access Error: 0: No error 1: Error	RW	0b1

Lockdown Control (LD_CTRL) register

The following table gives a bit-level description of the Lockdown Control register.

Table 11-60 LD_CTRL register

Bits	Name	Description	Type	Reset
[31:3]	-	Reserved	RO	0x0000_0000
[2]	LDI_ST	Lockdown interface status. Indicates the current value of the Lockdown interface: 0: Lockdown interface is de-asserted 1: Lockdown interface is asserted	RO	UNKNOWN
[1:0]	LOCK	Indicates the lock state of the Interrupt Router: 0b00: Open lockdown state. 0b01: Reserved and treated as 0b00 0b10: Partial lockdown state 0b11: Full lockdown state	RW	0b00

The LD_CTRL.LOCK field is not updateable, when the following are all true:
- Lockdown interface is asserted HIGH.
- Lockdown state of the Interrupt Router is Partial or Full.

Shared Interrupt Information (SHD_INT_INFO) register

The following table gives a bit-level description of the Shared Interrupt Information register.

Table 11-61 SHD_INT_INFO register

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RO	0x0000
[15:0]	ICI_DST	Interrupt Controller Destination. Each bit indicates whether the interrupt selected by the SHD_INT_SEL.INT_SEL field, can be routed to ICI interface associated with the bit, starting with bit 0 for ICI0 to bit 3 for ICI3: 0: Shared interrupt cannot be routed to the ICI{x} 1: Shared interrupt can be routed to the ICI{x} The value of this field is defined by the SI{x}_ICI_DST configuration option.	RO	CFG_DEF

Shared Interrupt Configuration (SHD_INT_CFG) register

The following table gives a bit-level description of the Shared Interrupt Configuration Register.

Table 11-62 SHD_INT_CFG register

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RO	0x0000
[15:0]	ICI_EN	Interrupt Controller Enable. Each bit selects whether the interrupt selected by the SHD_INT_SEL.INT_SEL field, is routed to the ICI interface associated with the bit, starting with bit 0 for ICI0 to bit 3 for ICI3: 0: Shared interrupt not routed to the ICI{x} 1: Shared interrupt routed to the ICI{x} Bits, where the respective bit in the SHD)_INT_INFO.ICI_DST field is 0b0 are Reserved and treated as RAZ/WI. The default value of this field is defined by the SI{x}_DEF_ICI configuration option.	RW	CFG_DEF

Shared Interrupt Lock Control (SHD_INT_LCTRL) register

The following table gives a bit-level description of the Shared Interrupt Lockdown Control register.

Table 11-63 SHD_INT_LCTRL register

Bits	Name	Description	Type	Reset
[31]	LOCK	Control the lock status of the interrupt selected by the SHD_INT_SEL.INT_SEL field: 0b0: Shared interrupt is not locked 0b1: Shared interrupt is locked This field becomes read-only when this field is set to 0b1 and the LD_CTRL.LOCK field is set to 0b10 or 0b11. For more information on the lock status, see *Appendix B Interrupt Router* on page Appx-B-274.	RW	0b0
[30:0]	-	Reserved	RO	0x0000_0000

Shared Interrupt Select (SHD_INT_SEL) register

The following table gives a bit-level description of the Shared Interrupt Select register.

Table 11-64 SHD_INT_SEL register

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RO	0x0000
[15:0]	INT_SEL	Selects which interrupt the SHD_INT_INFO, SHD_INT_CFG and SHD_INT_LCTRL registers refer to. When INT_SEL is greater than NUM_SHD_INT, the fields in SHD_INT_INFO, SHD_INT_CFG and SHD_INT_LCTRL are Reserved and treated as RAZ/WI.	RW	0x0000

Interrupt Router Tamper Status (INT_RTR_TMP_ST) register

The following table gives a bit-level description of the Interrupt Router Tamper Status register.

Table 11-65 INT_RTR_TMP_ST register

Bits	Name	Description	Type	Reset
[31]	TMP_ST_VLD	Indicates whether the INT_RTR_TMP_ST register contains valid data or not. 0: INT_RTR_TMP_ST does not contain valid data 1: INT_RTR_TMP_ST contains valid data This field is written 1 to clear, writing a value of 0 has no effect.	RW	0b0
[30]	TMP_ST_OVERFLW	Indicates whether a tamper transaction occurred, while the INT_RTR_TMP_ST.TMP_ST_VLD was 1: 0: No tamper transaction overflow occurred : Tamper transaction overflow occurred This field is written 1 to clear, writing a value of 0 has no effect.	RW	0b0
[29:12]	-	Reserved	RO	0x0_0000
[11:2]	TMP_TRANS_ADDR	Address of the register accessed by the tamper transaction. When TMP_ST_VLD is 0 this field is not valid.	RO	UNKNOWN
[1:0]	-	Reserved	RO	0x00

Note

This register can only be accessed by Secure privileged access from the Security Monitor. In the SSE-700 the Secure Enclave is the Security Monitor for the Interrupt Router. For more information, see *Appendix B Interrupt Router* on page Appx-B-274.

Interrupt Router Capability (INT_RTR_CAP) register

The following table gives a bit-level description of the Interrupt Router Capability register.

Table 11-66 INT_RTR_CAP register

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RO	0x0000
[3:0]	LDE_LVL	Level of the Lockdown Extension implemented by the Interrupt Router. Read as 0x2 – LDE.2 is implemented. All other values are Reserved.	RO	0x2

Interrupt Router Configuration (INT_RTR_CFG) register

The following table gives a bit-level description of the Interrupt Router Configuration register.

Table 11-67 INT_RTR_CFG Register

Bits	Name	Description	Type	Reset
[31:20]	-	Reserved	RO	0x000
[19:16]	NUM_ICI	Number of Interrupt Controllers Interrupt interface (ICI)s supported by the Interrupt Router: 0x3: 4 ICIs supported	RO	0x3
[15:0]	NUM_SHD_INT	Number of shared interrupts supported by the Interrupt Router: 0x0: 1 Shared interrupt 0x1: 2 Shared interrupts … 0x18B: 395 Shared interrupts	RO	CFG_DEF

Peripheral ID 4 (PID4) register

The following table gives a bit-level description of the Peripheral ID 4 register.

Table 11-68 PID4 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	Size	Number of 4KB blocks occupied by the System ID block. This field is deprecated.	RO	0x0
[3:0]	DES_2	JEP Continuation	RO	0x4

Peripheral ID 5 (PID5) register

The following table gives a bit-level description of the Peripheral ID 5 register.

Table 11-69 PID5 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 6 (PID6) register

The following table gives a bit-level description of the Peripheral ID 6 register.

Table 11-70 PID6 Register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 7 (PID7) register

The following table gives a bit-level description of the Peripheral ID 7 register.

Table 11-71 PID7 Register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 0 (PID0) register

The following table gives a bit-level description of the Peripheral ID 0 register.

Table 11-72 PID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PART_0	Bits [7:0] of part ID	RO	0x74

Peripheral ID 1 (PID1) register

The following table gives a bit-level description of the Peripheral ID 1 register.

Table 11-73 PID1 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	DES_0	Bits [3:0] of JEP 106 Identity	RO	0xB
[3:0]	PART_1	Bits [11:8] of part ID	RO	0x0

Peripheral ID 2 (PID2) register

The following table gives a bit-level description of the Peripheral ID 2 register.

Table 11-74 PID2 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	REVISION	Major revision of the System ID block	RO	0x0
[3]	JEDEC	Indicates the use of JEDEC JEP106 identification scheme	RO	0b1
[2:0]	DES_1	Bits [6:4] of JEP 106 Identity	RO	0b011

Peripheral ID 3 (PID3) register

The following table gives a bit-level description of the Peripheral ID 3 register.

Table 11-75 PID3 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	REVAND	Minor revision of the System ID block	RO	0x0
[3:0]	CMOD	Customer modification field	RO	0x0

Component ID 0 (CID0) register

The following table gives a bit-level description of the Component ID 0 register.

Table 11-76 CID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_0	Preamble 0	RO	0x0

Component ID 1 (CID1) register

The following table gives a bit-level description of the Component ID 1 register.

Table 11-77 CID1 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	CLASS	Class of the component	RO	0xF
[3:0]	PRMBL_1	Preamble 1	RO	0x0

Component ID 2 (CID2) register

The following table gives a bit-level description of the Component ID 2 register.

Table 11-78 CID2 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_2	Preamble 2	RO	0x05

Component ID 3 (CID3) register

The following table gives a bit-level description of the Component ID 3 register.

Table 11-79 CID3 Register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_3	Preamble 3	RO	0xB1

11.3.4 REFCLK Counter CNTControl register summary

This section summarizes the **REFCLK** Counter CNTControl registers.

REFCLK CNTControl is an implementation of a Memory Mapped Counter as defined by the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. The **REFCLK** Counter CNTControl and CNTRead frames are accessed at offset 0x1A20_0000 and 0x1A21_0000 respectively in the Host System memory map.

The CNTControl frame uses three additional undefined registers, for more information see the reference manuals stated above. The following table lists the additional three registers and their offset address in the CNTControl frame.

Table 11-80 REFCLK Counter register summary

Offset	Short name	Access	Name
0xC0	CNTSCR	RW	Counter Synchronization Control Register. *CNTControl Counter Synchronization Control (CNTSCR) register* on page 11-216
0xC4	-	RO	Reserved
0xC8	CNTSVL	RO	Counter Synchronized Counter Lower Value Register *CNTControl Synchronized Counter Value (CNTSVL) register* on page 11-216.
0xCC	CNTSVU	RO	Counter Synchronized Counter Upper Value Register. *CNTControl Synchronized Counter Value (CNTSVU) register* on page 11-217

CNTControl Counter Synchronization Control (CNTSCR) register

The following table gives a bit-level description of the REFCLK CNTControl Sync Control register.

Table 11-81 CNTSCR register

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved	RO	0x0000_0000
[0]	ENSYNC	Controls the way the Counter Control register EN bit operates: 0b0: The counter is enabled or disabled immediately 0b1: The enabling of the counter is delayed until just after the next rising edge at the **S32KCLK** Disabling the counter is not delayed.	RW	0b0

CNTControl Synchronized Counter Value (CNTSVL) register

The following table gives a bit-level description of the CNTControl Synchronized Counter Value Lower Word register.

Table 11-82 CNTSVL register

Bits	Name	Description	Type	Reset
[31:0]	CNTSVL	**S32KCLK**-sampled value of counter, lower word – CNTSV[31:0]	RO	0x0000_0000

CNTControl Synchronized Counter Value (CNTSVU) register

The following table gives a bit-level description of the Synchronized Counter Value Upper Word register.

Table 11-83 CNTSVU register

Bits	Name	Description	Type	Reset
[31:0]	CNTSVU	Reads back the value of the counter sampled on the rising of edge of **S32KCLK**. Returns the upper word of CNTSV[63:32].	RO	0x0000_0000

11.3.5 System ID register summary

This section summarizes the System ID registers.

The System ID block registers identify:

- SoC and its implementor
- Version and implementor of SSE-700

The registers are in the AONTOP domain and are accessed at offset 0x1A00_0000 in the Host System memory map.

Table 11-84 System ID register summary

Offset	Short name	Access	Name
0x000	BSYS_CFG0	RO	Base System Configuration 0 *Base System Config 0 (BSYS_CFG0) register* on page 11-218
0x004	BSYS_CFG1	RO	Base System Configuration 1 *Base System Config 1 (BSYS_CFG1) register* on page 11-218
0x008	BSYS_CFG2	RO	Base System Configuration 2 *Base System Config 2 (BSYS_CFG2) register* on page 11-219
0x00C	BSYS_CFG3	RO	Base System Configuration 3 *Base System Config 3 (BSYS_CFG3) register* on page 11-220
0x010 – 0x03C	-	RO	Reserved
0x040	SOC_ID	RO	SoC Identification *SoC Identification (SOC_ID) register* on page 11-220
0x044 – 0xFC4	-	RO	Reserved
0xFC8	IIDR	RO	Implementer Identification Register *Implementer Identification Register (IIDR)* on page 11-221
0xFCC	-	RO	Reserved
0xFD0	PID4	RO	Peripheral ID4 *Peripheral ID 4 (PID4) register* on page 11-221
0xFD4	PID5	RO	Peripheral ID5 *Peripheral ID 5 (PID5) register* on page 11-222
0xFD8	PID6	RO	Peripheral ID6 *Peripheral ID 6 (PID6) register* on page 11-222
0xFDC	PID7	RO	Peripheral ID7 *Peripheral ID 7 (PID7) register* on page 11-222
0xFE0	PID0	RO	Peripheral ID0 *Peripheral ID 0 (PID0) register* on page 11-222

Table 11-84 System ID register summary (continued)

Offset	Short name	Access	Name
0xFE4	PID1	RO	Peripheral ID1 *Peripheral ID 1 (PID1) register* on page 11-222
0xFE8	PID2	RO	Peripheral ID2 *Peripheral ID 2 (PID2) register* on page 11-222
0xFEC	PID3	RO	Peripheral ID3 *Peripheral ID 3 (PID3) register* on page 11-223
0xFF0	CID0	RO	Component ID0 *Component ID 0 (CID0) register* on page 11-223
0xFF4	CID1	RO	Component ID1 *Component ID 1 (CID1) register* on page 11-223
0xFF8	CID2	RO	Component ID2 *Component ID 2 (CID2) register* on page 11-223
0xFFC	CID3	RO	Component ID3 *Component ID 3 (CID3) register* on page 11-224

Note

The System ID registers support only 32-bit word-aligned access. Any access by other bit size or unaligned access is not supported. For any non 32-bit word aligned access, the following happens:

- Generates an error and is treated as RAZ/WI

The System ID registers have the following behavior:

- Any read to a write-only register is treated as RAZ
- Any write to a read-only register is treated as WI

In both cases no error is generated.

Base System Config 0 (BSYS_CFG0) register

The following table gives a bit-level description of the Base System Config 0 register.

Table 11-85 BSYS_CFG0 register

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RO	0x000_0000
[3:0]	NUM_HOST_CPU	Number of Host processor cores used	RO	HOST_CPU_NUM_CORES

Base System Config 1 (BSYS_CFG1) register

The following table gives a bit-level description of the Base System Config 1 register.

Table 11-86 BSYS_CFG1 register

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RO	0x0000
[15:12]	EXT_SYS3	External System 3 Configuration: 0x0: External System 3 not implemented 0x1: Reserved 0x2: External System 3 implemented without Arm TrustZone 0x3: External System 3 implemented with Arm TrustZone All other values are Reserved. For SSE-700 this field always reads as 0x0.	RO	0x0
[11:8]	EXT_SYS2	External System 2 Configuration: 0x0: External System 2 not implemented 0x1: Reserved 0x2: External System 2 implemented without Arm TrustZone 0x3: External System 2 implemented with Arm TrustZone All other values are Reserved. For SSE-700 this field always reads as 0x0.	RO	0x0
[7:4]	EXT_SYS1	External System 1 Configuration: 0x0: External System 1 not implemented. 0x1: Reserved 0x2: External System 1 implemented without Arm TrustZone 0x3: External System 1 implemented with Arm TrustZone All other values are Reserved. For SSE-700 this field always reads as 0x3.	RO	0x3
[3:0]	EXT_SYS0	External System 0 Configuration: 0x0: External System 0 not implemented. 0x1: Reserved 0x2: External System 0 implemented without Arm TrustZone 0x3: External System 0 implemented with Arm TrustZone All other values are Reserved. For SSE-700 this field always reads as 0x3.	RO	0x3

Base System Config 2 (BSYS_CFG2) register

The following table gives a bit-level description of the Base System Config 2 register.

Table 11-87 BSYS_CFG2 register

Bits	Name	Description	Type	Reset
[31:25]	-	Reserved	RO	`0x00`
[24]	OCVM_EN	Indicates if the OCVM interface is supported: `0b0`: OCVM interface is not supported. `0b1`: OCVM interface is supported. For SSE-700 this field always reads as `0b1`.	RO	`0x1`
[23:20]	NUM_EXP_MST	Number of Expansion Master interfaces: `0x0`: No Expansion Master interfaces. `0x1`: 1 Expansion Master interface `0x2`: 2 Expansion Master interfaces `0x3`: 3 Expansion Master interfaces `0x4`: 4 Expansion Master interfaces All other values are Reserved. For SSE-700 this field always reads as `0x2`.	RO	`0x2`
[19:16]	NUM_EXP_SLV	Number of Expansion Slave interfaces: `0x0`: No Expansion Slave interfaces `0x1`: 1 Expansion Slave interface `0x2`: 2 Expansion Slave interfaces `0x3`: 3 Expansion Slave interfaces `0x4`: 4 Expansion Slave interfaces All other values are Reserved. For SSE-700 this field always reads as `0x2`.	RO	`0x2`
[15:0]	-	Reserved	RO	`0x0000`

Base System Config 3 (BSYS_CFG3) register

The following table gives a bit-level description of the Base System Config 3 register.

Table 11-88 BSYS_CFG3 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	`0x0000_0000`

SoC Identification (SOC_ID) register

The following table gives a bit-level description of the SoC Identification register.

Table 11-89 SOC_ID register

Bits	Name	Description	Type	Reset
[31:20]	PRODUCT_ID	User defined value identifying the SoC. This field is set to **SOCPRTID**.	RO	CFG_DEF
[19:16]	VARIANT	User defined value variant or major revision of the SoC. This field is set to **SOCVAR**.	RO	CFG_DEF
[15:12]	REVISION	Use defined value used to distinguish minor revisions of the SoC. This field is set to **SOCREV**.	RO	CFG_DEF
[11:0]	IMPLEMENTER	Contains the JEP106 code of the company that used the SoC: [11:8] JEP106 continuation code of implementer. Provided by bits 10:7 of **SOCIMPLID**. [7] Always 0. [6:0] JEP106 identity code of implementer. Provided by bits 6:0 of **SOCIMPLID**	RO	CFG_DEF

Implementer Identification Register (IIDR)

The following table gives a bit-level description of the Implementer Identification Register.

Table 11-90 IIDR register

Bits	Name	Description	Type	Reset
[31:20]	PRODUCT_ID	Product ID of SSE-700	RO	`0x749`
[19:16]	VARIANT	Variant or major revision of SSE-700	RO	`0x1`
[15:12]	REVISION	Minor revisions of SSE-700	RO	`0x0`
[11:0]	IMPLEMENTER	Contains the JEP106 code of the company that implemented SSE-700: [11:8] JEP106 continuation code of implementer [7] Always 0 [6:0] JEP106 identity code of implementer	RO	`0x43B`

Peripheral ID 4 (PID4) register

The following table gives a bit-level description of the Peripheral ID 4 register.

Table 11-91 PID4 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	`0x00_0000`
[7:4]	Size	Number of 4KB occupied by the System ID block. This field is deprecated.	RO	`0x0`
[3:0]	DES_2	JEP Continuation	RO	`0x4`

Peripheral ID 5 (PID5) register

The following table gives a bit-level description of the Peripheral ID 5 register.

Table 11-92 PID5 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 6 (PID6) register

The following table gives a bit-level description of the Peripheral ID 6 register.

Table 11-93 PID6 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 7 (PID7) register

The following table gives a bit-level description of the Peripheral ID 7 register.

Table 11-94 PID7 Register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 0 (PID0) register

The following table gives a bit-level description of the Peripheral ID 0 register.

Table 11-95 PID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PART_0	Bits [7:0] of part ID	RO	0x49

Peripheral ID 1 (PID1) register

The following table gives a bit-level description of the Peripheral ID 1 register.

Table 11-96 PID1 Register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	DES_0	Bits [3:0] of JEP 106 Identity	RO	0xB
[3:0]	PART_1	Bits [11:8] of part ID	RO	0x7

Peripheral ID 2 (PID2) register

The following table gives a bit-level description of the Peripheral ID 2 register.

Table 11-97 PID2 Register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	REVISION	Major revision of the System ID block	RO	0x1
[3]	JEDEC	Indicates the use of JEDEC JEP106 identification scheme	RO	0b1
[2:0]	DES_1	Bits [6:4] of JEP 106 Identity	RO	0b011

Peripheral ID 3 (PID3) register

The following table gives a bit-level description of the Peripheral ID 3 register.

Table 11-98 PID3 Register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	REVAND	Minor revision of the System ID block	RO	0x0
[3:0]	CMOD	Customer modification field	RO	0x0

Component ID 0 (CID0) register

The following table gives a bit-level description of the Component ID 0 register.

Table 11-99 CID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_0	Preamble 0	RO	0x0D

Component ID 1 (CID1) register

The following table gives a bit-level description of the Component ID 1 register.

Table 11-100 CID1 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	CLASS	Class of the component	RO	0xF
[3:0]	PRMBL_1	Preamble 1	RO	0x0

Component ID 2 (CID2) register

The following table gives a bit-level description of the Component ID 2 register.

Table 11-101 CID2 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_2	Preamble 2	RO	0x05

Component ID 3 (CID3) register

The following table gives a bit-level description of theComponent ID 3 register.

Table 11-102 CID3 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_3	Preamble 3	RO	0xB1

11.3.6 Boot register aummary

This section summarizes the Boot Register registers.

The Boot Registers are accessed at offset 0x0000_0000 in the Host System memory map. The Boot Registers are written once only by the Secure Enclave. The Boot Register is in the AONTOP power domain.

Table 11-103 Boot register summary

Offset	Short name	Access	Name
0x000 – 0x00C	BIR{0-3}	RW	Boot Instruction Register {0-3} *Boot Instruction Register {0-3} (BIR{0-3})* on page 11-225
0x010 – 0x03C	BIR{4-15}	RO	Boot Instruction Register {4-15} *Boot Instruction Register {4-15} (BIR{4-15})* on page 11-225
0x040 – 0xFCC	-	RO	Reserved
0xFD0	PID4	RO	Peripheral ID4 *Peripheral ID 4 (PID4) register* on page 11-225
0xFD4	PID5	RO	Peripheral ID5 *Peripheral ID 5 (PID5) register* on page 11-226
0xFD8	PID6	RO	Peripheral ID6 *Peripheral ID 6 (PID6) register* on page 11-225
0xFDC	PID7	RO	Peripheral ID7 *Peripheral ID 7 (PID7) register* on page 11-226
0xFE0	PID0	RO	Peripheral ID0 *Peripheral ID 0 (PID0) register* on page 11-226
0xFE4	PID1	RO	Peripheral ID1 *Peripheral ID 1 (PID1) register* on page 11-226
0xFE8	PID2	RO	Peripheral ID2 *Peripheral ID 2 (PID2) register* on page 11-226
0xFEC	PID3	RO	Peripheral ID3 *Peripheral ID 3 (PID3) register* on page 11-227
0xFF0	CID0	RO	Component ID0 *Component ID 0 (CID0) register* on page 11-227
0xFF4	CID1	RO	Component ID1*Component ID 1 (CID1) register* on page 11-227
0xFF8	CID2	RO	Component ID2 *Component ID 2 (CID2) register* on page 11-227
0xFFC	CID3	RO	Component ID3 *Component ID 3 (CID3) register* on page 11-227

Note

An error is generated if any of the following occur:

- Any attempt to write to the register from any master other than the Secure Enclave.
- Secure Enclave writes to the same register more than once.
- Any attempt to write to a read-only register, by any master.
- Any attempt to read a Reserved register.

The registers of the Boot Register support the following accesses:

- Read accesses support any size access, but must be aligned to the size of the access.
- Write accesses must be 32-bit word-aligned accesses.
- Any access not meeting these requirements generates an error and is treated as RAZ/WI.

Boot Instruction Register {0-3} (BIR{0-3})

The following table gives a bit-level description of the Boot Instruction Register {0-3}.

Table 11-104 BIR{0-3} register

Bits	Name	Description	Type	Reset
[31:0]	BOOT_INSTR	Write-once register. Sets the instructions for the Host processor.	RW	UNKNOWN

Note

This is a write-once register by the Secure Enclave.

Boot Instruction Register {4-15} (BIR{4-15})

The following table gives a bit-level description of the Boot Instruction Register {4-15}.

Table 11-105 BIR{4-15} register

Bits	Name	Description	Type	Reset
[31:0]	BOOT_INSTR	Reads of this address return the value in the Boot Instruction Register {0-3} registers: BIR{4,8,12} return the value in BIR0 BIR{5,9,13} return the value in BIR1 BIR{6,10,14} return the value in BIR2 BIR{7,11,15} return the value in BIR3	RO	UNKNOWN

Peripheral ID 4 (PID4) register

The following table gives a bit-level description of the Peripheral ID 4 register.

Table 11-106 PID4 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	Size	Number of 4KB blocks occupied by the System ID block. This field is deprecated.	RO	0x0
[3:0]	DES_2	JEP Continuation	RO	0x4

Peripheral ID 6 (PID6) register

The following table gives a bit-level description of the Peripheral ID 6 register.

Table 11-107 PID6 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 7 (PID7) register

The following table gives a bit-level description of the Peripheral ID 7 register.

Table 11-108 PID7 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 5 (PID5) register

The following table gives a bit-level description of the Peripheral ID 5 register.

Table 11-109 PID5 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 0 (PID0) register

The following table gives a bit-level description of the Peripheral ID 0 register.

Table 11-110 PID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PART_0	Bits [7:0] of part ID	RO	0x72

Peripheral ID 1 (PID1) register

The following table gives a bit-level description of the Peripheral ID 1 register.

Table 11-111 PID1 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	DES_0	Bits [3:0] of JEP 106 Identity	RO	0xB
[3:0]	PART_1	Bits [11:8] of part ID	RO	0x0

Peripheral ID 2 (PID2) register

The following table gives a bit-level description of the Peripheral ID 2 register.

Table 11-112 PID2 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	REVISION	Major revision of the System ID bloc	RO	0x0

Table 11-112 PID2 register (continued)

Bits	Name	Description	Type	Reset
[3]	JEDEC	Indicates the use of JEDEC JEP106 identification scheme	RO	0b1
[2:0]	DES_1	Bits [6:4] of JEP 106 Identity.	RO	0b011

Peripheral ID 3 (PID3) register

The following table gives a bit-level description of the Peripheral ID 3 register.

Table 11-113 PID3 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	REVAND	Minor revision of the System ID block	RO	0x0
[3:0]	CMOD	Customer modification field	RO	0x0

Component ID 0 (CID0) register

The following table gives a bit-level description of the Component ID 0 register.

Table 11-114 CID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_0	Preamble 0	RO	0x0D

Component ID 1 (CID1) register

The following table gives a bit-level description of the Component ID 1 register.

Table 11-115 CID1 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	CLASS	Class of the component	RO	0xF
[3:0]	PRMBL_1	Preamble 1	RO	0x0

Component ID 2 (CID2) register

The following table gives a bit-level description of the Component ID 2 register.

Table 11-116 CID2 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_2	Preamble 2	RO	0x05

Component ID 3 (CID3) register

The following table gives a bit-level description of the Component ID 3 register.

Table 11-117 CID3 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_3	Preamble 3	RO	0xB1

11.3.7 GPIO Control register summary

This section summarizes the GPIO Control registers.

Table 11-118 GPIO Control register summary

Offset	Access type	Name	Short name
0x000 – 0x07C	RW	General Purpose Output {0-31} Control *General Purpose Output {0-31} Control (GPO{0-31}_CTRL) register* on page 11-229	GPO{0-31}_CTRL
0x080 – 0x0FC	RO	Reserved	-
0x100 – 0x17C	RW	General Purpose Input {0-31} *General Purpose Input {0-31} Status (GPI{0-31}_ST) register* on page 11-229	GPI{0-31}_ST
0x180 – 0xEFC	RO	Reserved	-
0xF00	RW	Integration Mode Control *Integration Mode Control register* on page 11-230	ITCTRL
0xF0 - 0xF9C	RO	Reserved	-
0xFA0	RW	Claim Tag Set *Claim Tag Set register* on page 11-230	CLAIMSET
0xFA4	RW	Claim Tag Clear *Claim Tag Clear register* on page 11-230	CLAIMCLR
0xFA8	RO	Device Affinity 0 *Device Affinity 0 register* on page 11-230	DEVAFF0
0xFAC	RO	Device Affinity 1 *Device Affinity 1 register* on page 11-230	DEVAFF1
0xFB0	WO	Software Lock Access *Software Lock Access register* on page 11-231	LAR
0xFB4	RO	Software Lock Status *Software Lock Status register* on page 11-231	LSR
0xFB8	RO	Authentication Status *Authentication Status (AUTHSTATUS) register* on page 11-231	AUTHSTATUS
0xFBC	RO	Device Architecture *Device Architecture (DEVARCH) register* on page 11-233	DEVARCH
0xFC0	RO	Device Configuration 2 *Device Configuration 2 (DEVID2) register* on page 11-233	DEVID2
0xFC4	RO	Device Configuration 1 *Device Configuration 1 (DEVID1) register* on page 11-233	DEVID1
0xFC8	RO	Device Configuration *Device Configuration (DEVID) register* on page 11-233	DEVID
0xFCC	RO	Device Type Identifier *Device Type Identifier (DEVTYPE) register* on page 11-234	DEVTYPE
0xFD0	RO	Peripheral ID4 *Peripheral ID 4 (PID4) register* on page 11-234	PID4
0xFD4	RO	Peripheral ID5 *Peripheral ID 5 (PID5) register* on page 11-235	PID5
0xFD8	RO	Peripheral ID6 *Peripheral ID 6 (PID6) register* on page 11-235	PID6
0xFDC	RO	Peripheral ID7 *Peripheral ID 7 (PID7) register* on page 11-235	PID7
0xFE0	RO	Peripheral ID0 *Peripheral ID 0 (PID0) register* on page 11-235	PID0
0xFE4	RO	Peripheral ID1 *Peripheral ID 1 (PID1) register* on page 11-235	PID1

Table 11-118 GPIO Control register summary (continued)

Offset	Access type	Name	Short name
0xFE8	RO	Peripheral ID2 *Peripheral ID 2 (PID2) register* on page 11-236	PID2
0xFEC	RO	Peripheral ID3 *Peripheral ID 3 (PID3) register* on page 11-236	PID3
0xFF0	RO	Component ID0 *Component ID 0 (CID0) register* on page 11-236	CID0
0xFF4	RO	Component ID1 *Component ID 1 (CID1) register* on page 11-236	CID1
0xFF8	RO	Component ID2 *Component ID 2 (CID2) register* on page 11-237	CID2
0xFFC	RO	Component ID3 *Component ID 3 (CID3) register* on page 11-237	CID3

Note

The GPIO Control registers only support 32-bit word aligned accesses. Any access of other size or unaligned access is not supported. For any non 32-bit aligned word access, the resulting behavior is:

- Generates an error and is treated as RAZ/WI.

The GPIO Control registers have the following behavior:

- Any read to a write-only register is treated as RAZ.
- Any write to a read-only register is treated as WI.

In both cases, no error is generated.

General Purpose Output {0-31} Control (GPO{0-31}_CTRL) register

The following table gives a bit-level description of the General Purpose Output {0-31} Control register.

Table 11-119 GPO{0-31}_CTRL register

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved	RO	0x0000_0000
[0]	GPO	Controls the value of the general-purpose output. When AUTHSTATUS.NSID is 0b10 it is IMPLEMENTATION DEFINED whether this field behaves as follows: Is treated as RAZ/WI with the respective GPO output driven to 0b0. Is treated a read-write, but with the respective GPO output driven to 0b0. For SSE-700 this field is Reserved and treated as RAZ/WI for GPO{x}_CTRL registers where x is greater than or equal to 1.	RW	0b0

General Purpose Input {0-31} Status (GPI{0-31}_ST) register

The following table gives a bit-level description of the General Purpose Input {0-31} Status register.

Table 11-120 GPI{0-31}_ST register

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved	RO	0x0000_0000
[0]	GPI	Provides the status of the General Purpose input. When AUTHSTATUS.NSID is 0b10 the value in this field is always 0b0. For SSE-700 this field is Reserved and treated as RAZ/WI for all GPI{x}_ST registers.	RO	0b0

Integration Mode Control register

The following table gives a bit-level description of the Integration Mode Control register.

Table 11-121 Integration Mode Control register

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved	RO	0x0000_0000
[0]	IME	Integration Mode Enable: 0b0: Component must enter functional mode. 0b1: Component must enter integration mode. The GPIO does not support the integration mode and therefore this field ignores writes which set this field to 0b1.	RW	0b0

Claim Tag Set register

The following table gives a bit-level description of the Claim Tag Set register.

Table 11-122 Claim Tag Set register

Bits	Name	Description	Type	Reset
[31:0]	SET	The GPIO does not support and Claim Tags. Therefore, all bits are treated as RAZ/WI.	RW	0x0000_0000

Claim Tag Clear register

The following table gives a bit-level description of the Claim Tag Clear register.

Table 11-123 Claim Tag Clear register

Bits	Name	Description	Type	Reset
[31:0]	CLR	The GPIO does not support and Claim Tags. Therefore, all bits are treated as RAZ/WI.	RW	0x0000_0000

Device Affinity 0 register

The following table gives a bit-level description of the Device Affinity 0 register.

Table 11-124 Device Affinity 0 register

Bits	Name	Description	Type	Reset
[31:0]	DEVAFF0	The field reads as zero	RO	0x0000_0000

Device Affinity 1 register

The following table gives a bit-level description of the Device Affinity 1 register.

Table 11-125 Device Affinity 1 register

Bits	Name	Description	Type	Reset
[31:0]	DEVAFF1	This field reads as zero	RO	0x0000_0000

Software Lock Access register

The following table gives a bit-level description of the Software Lock Access register.

Table 11-126 Software Lock Access register

Bits	Name	Description	Type	Reset
[31:0]	KEY	Key value. The GPIO Control does not implement the software lock mechanism so writes to this field are treated as WI. Reads to this field return 0x0000_0000.	WO	0x0000_0000

Software Lock Status register

The following table gives a bit-level description of the Software Lock Status register.

Table 11-127 Software Lock Status register

Bits	Name	Description	Type	Reset
[31:3]	-	Reserved	RO	0x0000_0000
[2]	nTT	This field always reads as 0 which indicates a 32-bit Software Lock Access register is implemented.	RO	0b0
[1]	SLK	Software lock status	RO	0b0
[0]	SLI	Software lock mechanism is implemented. This field always reads 0b0 as the GPIO Control does not implement the software lock mechanism.	RO	0b0

Authentication Status (AUTHSTATUS) register

The following table gives a bit-level description of the Authentication Status register.

Table 11-128 AUTHSTATUS register

Bits	Name	Description	Type	Reset
[31:12]	-	Reserved	RO	0x00_0000
[11:10]	HNID	Hypervisor non-invasive debug. 0b00: Functionality not implemented or controlled elsewhere 0b01: Reserved 0b10: Functionality disabled 0b11: Functionality enabled This field always reads as 0b00 for the GPIO.	RO	0b00

Table 11-128 AUTHSTATUS register (continued)

Bits	Name	Description	Type	Reset
[9:8]	HID	Hypervisor invasive debug: `0b00`: Functionality not implemented or controlled elsewhere `0b01`: Reserved `0b10`: Functionality disabled `0b11`: Functionality enabled This field always reads as `0b00` for the GPIO.	RO	`0b00`
[7:6]	SNID	Secure non-invasive debug: `0b00`: Functionality not implemented or controlled elsewhere `0b01`: Reserved `0b10`: Functionality disabled `0b11`: Functionality enabled This field always reads as `0b00` for the GPIO.	RO	`0b00`
[5:4]	SID	Secure invasive debug: `0b00`: Functionality not implemented or controlled elsewhere `0b01`: Reserved `0b10`: Functionality disabled `0b11`: Functionality enabled This field always reads as `0b00` for the GPIO.	RO	`0b00`
[3:2]	NSNID	Non-secure non-invasive debug: `0b00`: Functionality not implemented or controlled elsewhere `0b01`: Reserved `0b10`: Functionality disabled `0b11`: Functionality enabled This field always reads as `0b00` for the GPIO.	RO	`0b00`
[1:0]	NSID	Non-secure invasive debug: `0b00`: Functionality not implemented or controlled elsewhere `0b01`: Reserved `0b10`: Functionality disabled `0b11`: Functionality enabled This field can take the following values: `0b10`: when **DBGEN** is 0 `0b11`: when **DBGEN** is 1 The reset value of this field depends on the value of **DBGEN** input.	RO	See description.

Device Architecture (DEVARCH) register

The following table gives a bit-level description of the Device Architecture register.

Table 11-129 DEVARCH register

Bits	Name	Description	Type	Reset
[31:21]	ARCHITECT	Defines the architect of the component: Bits[31:28] Indicates the JEP106 continuation code Bits[27:21] Indicates the JEP106 identification code See the Standard Manufacturers Identification Code for information about JEP106. For components where ARM is the architect, this 11-bit field returns `0x23B`.	RO	`0x000`
[20]	PRESENT	Indicates the presence of this register: `0b0`: DEVARCH is not present. Bits[31:0] must be RAZ. `0b1`: DEVARCH is present. This field always reads as `0b0` for the GPIO	RO	`0b0`
[19:16]	REVISION	Architecture revision. Returns the revision of the architecture that the ARCHID field specifies.	RO	`0x0`
[15:0]	ARCHID	Architecture ID. Returns a value that identifies the architecture of the component.	RO	`0x0000`

Device Configuration 2 (DEVID2) register

The following table gives a bit-level description of the Device Configuration 2 register.

Table 11-130 DEVID2 Register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	`0x0000_0000`

Device Configuration 1 (DEVID1) register

The following table gives a bit-level description of the Device Configuration 1 register.

Table 11-131 DEVID1 Register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	`0x0000_0000`

Device Configuration (DEVID) register

The following table gives a bit-level description of the Device Configuration register.

Table 11-132 DEVID register

Bits	Name	Description	Type	Reset
[31:14]	-	Reserved	RO	0_0000
[13:8]	NUM_GPI	Number of GPI{x}_ST registers which are implemented: 0x00: No GPI{x}_ST register are implemented 0x01: 1 GPI{x}_ST register is implemented, starting with GPI0_ST 0x02: 2 GPI{x}_ST registers are implemented, starting with GPI0_ST up to GPI1_ST … 0x20: 32 GPI{x}_ST registers are implemented, starting with GPI0_ST up to GPI31_ST For SSE-700 this field always reads as 0x00.	RO	0x00
[7:6]	-	Reserved	RO	0b00
[5:0]	NUM_GPO	Number of GPO{x}_CTRL registers which are implemented: 0x00: No GPO{x}_CTRL register are implemented 0x01: 1 GPO{x}_CTRL register is implemented, starting with GPO0_CTRL 0x02: 2 GPO{x}_CTRL registers are implemented, starting with GPO0_CTRL up to GPO1_CTRL … 0x20: 32 GPO{x}_CTRL registers are implemented, starting with GPO0_CTRL up to GPO31_CTRL For SSE-700 this field always reads as 0x01.	RO	0x01

Device Type Identifier (DEVTYPE) register

The following table gives a bit-level description of the Device Type Identifier register.

Table 11-133 DEVTYPE register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	SUB	Sub type for the component device type: 0x0: Indicates Other, undefined	RO	0x0
[3:0]	MAJOR	Major type for the component device type: 0x0: Indicates Miscellaneous	RO	0x0

Peripheral ID 4 (PID4) register

The following table gives a bit-level description of the Peripheral ID 4 register.

Table 11-134 PID4 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	Size	Number of 4KB units occupied by the System ID block. This field is deprecated.	RO	0x0
[3:0]	DES_2	JEP Continuation	RO	0x4

Peripheral ID 5 (PID5) register

The following table gives a bit-level description of the Peripheral ID 5 register.

Table 11-135 PID5 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved.	RO	0x0000_0000

Peripheral ID 6 (PID6) register

The following table gives a bit-level description of the Peripheral ID 6 register.

Table 11-136 PID6 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 7 (PID7) register

The following table gives a bit-level description of the Peripheral ID 7 register.

Table 11-137 PID7 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 0 (PID0) register

The following table gives a bit-level description of the Peripheral ID 0 register.

Table 11-138 PID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PART_0	Bits [7:0] of part ID	RO	0xF0

Peripheral ID 1 (PID1) register

The following table gives a bit-level description of the Peripheral ID 1 register.

Table 11-139 PID1 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	DES_0	Bits [3:0] of JEP 106 Identity	RO	0xB
[3:0]	PART_1	Bits [11:8] of part ID	RO	0x9

Peripheral ID 2 (PID2) register

The following table gives a bit-level description of the Peripheral ID 2 register.

Table 11-140 PID2 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	REVISION	Major revision of the System ID block	RO	0x0
[3]	JEDEC	Indicates the use of JEDEC JEP106 identification scheme	RO	0b01
[2:0]	DES_1	Bits [6:4] of JEP 106 Identity	RO	0b011

Peripheral ID 3 (PID3) register

The following table gives a bit-level description of the Peripheral ID 3 register.

Table 11-141 PID3 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	REVAND	Minor revision of the System ID block	RO	0x0
[3:0]	CMOD	Customer modification field	RO	0x0

Component ID 0 (CID0) register

The following table gives a bit-level description of the Component ID 0 register.

Table 11-142 CID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_0	Preamble 0	RO	0x0D

Component ID 1 (CID1) register

The following table gives a bit-level description of the Component ID 1 register.

Table 11-143 CID1 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	CLASS	Class of the component	RO	0x9
[3:0]	PRMBL_1	Preamble 1	RO	0x0

Component ID 2 (CID2) register

The following table gives a bit-level description of the Component ID 2 register.

Table 11-144 CID2 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_2	Preamble 2	RO	0x05

Component ID 3 (CID3) register

The following table gives a bit-level description of the Component ID 3 register.

Table 11-145 CID3 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_3	Preamble 3	RO	0xB1

Chapter 12
Software sequences

This chapter describes the software sequences that control functions within the SSE-700.

In this section, there are details of software sequences that control or perform certain functions in SSE-700 subsystem.

It contains the following sections:

12.1 Power control

This section shows example software sequences for controlling the power modes of the domains within SSE-700.

Arm recommends that these examples are used as starting points for development of SoC firmware.

Arm recommends that all power control sequences are performed by Secure software, either on the Secure Enclave or the Host CPU.

In all software sequences for power control, software is responsible for ensuring that all clocks used within the domain, and by the power control logic, are configured to use a clock source which is available in the current power state of the SoC. Software may be required to change the clock source before starting a power sequence. For example, if the clock source for the SYSPLL is in the SYSTOP domain, whenever the SYSTOP domain enters the OFF or MEM_RET power-modes, the SYSPLL source will no longer be available. Software is required to switch any clock source that is currently using SYSPLL to another clock source before allowing SYSTOP to enter the OFF or MEM_RET power-mode. Software can then re-enable the use of the SYSPLL when the SYSTOP domain enters the FUNC_RET or ON power mode and the SYSPLL output has been locked.

This section contains the following subsections:

12.1.1 CORE{0-3} and CLUSTOP power domains

This section describes the software sequences controlling the CORE{0-3} and CLUSTOP power domains.

The *Power Control System Architecture* describes the software sequences for the CORE{0-3} and CLUSTOP power domains, with the following exceptions or additional requirements:

- There is no *System Control Processor* (SCP), and the expected behaviour is the PPU policy and is left at its default value of dynamic OFF.
- Execution of a WFE instruction does not cause entry into the FULL_RET power mode for a core.
- Before the last Host CPU core enters the OFF-power mode software must:
  - Save the configuration of the Host GIC if the HOST_CPU_CLUS_PWR_REQ.PWR_REQ is set to `0b0`
  - If the BSYS_PWR_REQ.SYSTOP_PWR_REQ field is `0b000` or `0b001`, it must save the configuration of components in the SYSTOP domain
  - If the BSYS_PWR_REQ.REFCLK_REQ field is `0b0`, it must move any future **REFCLK** time events to the S32K time domain

The selection of the power mode, which the CLUSTOP power domain enters when it is quiescent and all CORE{0-3} domains are in OFF, is set by the HOST_CPU_CLUS_PWR_REQ register. Arm recommends that software sets this register when the last Host CPU core is entering the OFF power mode.

Other things to consider before the last Host CPU enters the OFF-power mode:

- Software should configure the wakeup conditions. For example, using the **REFCLK** timer 0 to generate an interrupt at a point in time:
  - If the software does not want the Host CPU to wake up when CLUSTOP is in OFF or MEM_RET power-mode, it must set the Host Base System Control BSYS_PWR_REQ.WAKEUP_EN bit to `0b0`.
- The response time of the system to wakeup events. This determines the minimum power states of the SSE-700 subsystem and the allowed power modes its domains can enter.

12.1.2 SYSTOP power domains

The SYSTOP power domain is shared between all systems within SSE-700 subsystem.

Both the Secure Enclave and the Host System Base System Control registers contain the BSYS_PWR_REQ.SYSTOP_PWR_REQ field which enables software to request the minimal power mode of the SYSTOP domain.

The hardware automatically selects the power-mode which the SYSTOP domain enters based on hardware indicators and the values in the BSYS_PWR_REQ.SYSTOP fields. The hardware selects the minimum power mode which meets all the requirements. Software must be able to handle the case where SYSTOP does not enter the power-mode requested in the BSYS_PWR_REQ.SYSTOP field.

Software must do the following:

- Before the software attempts to access the volatile memory, it must set the BSYS_PWR_REQ.SYSTOP_PWR_REQ to a value other than `0b000`. Failure to do so can lead to loss of data when there are no outstanding transactions in the SYSTOP domain.
- Before the software sets the BSYS_PWR_REQ.SYSTOP_PWR_REQ field to a value where there could be loss of data, software makes sure that it has saved all required information. Software is then responsible for restoring the information when it sets the field to a value where the data cannot be loss.

Arm recommends the following for controlling the SYSTOP power domain:

- Software does not change the default PPU_PWPR policy register for the SYSTOP PPU.
- Software uses the BSYS_PWR_REQ registers in the Host or Secure Enclave Base System Control blocks to control the power mode of the domain.
- During boot of the SoC, software sets the BSYS_PWR_REQ.SYSTOP_PWR_REQ to a value other than `0b000`. Software does not change this value unless it no longer requires the contents of the volatile memory.
- The BSYS_PWR_REQ register of the Host System Base System Control registers is only accessed by one agent, either secure software executing on the Host CPU or software executing on the Secure Enclave. Any request to make changes to the power modes of the External System are made via messages sent to either the Host CPU or Secure Enclave using the MHUs in the system. It is not required for the External System to send a message when it needs access to resources inside SYSTOP, but only when it requires that SYSTOP does not enter a specific power mode were lost of data could occur.
- Software considers the implications of powering down of the SYSTOP domain. For example, if **SYSPLL** is in the SYSTOP domain then any clocks which are generated from **SYSPLL** are switched to another clock source.
- Software programs the entry delay values of the SYSTOP PPU to a value which meets the requirements of the current operating conditions of the SoC. By default, SYSTOP PPU places the SYSTOP domain into the lowest power mode when all indications, both hardware and software, reveal that there is no requirement for SYSTOP to be in the ON power mode. This can impact the performance if the traffic pattern of accesses to the SYSTOP domain is regular, but has a large period in between. This can be the case where the External System is only issuing a single access at a time to the Host System.

Note

This section is written with the expectation that the SYSTOP PPU power policy is set to dynamic OFF. Programming of another policy may lead to different behaviour.

12.1.3 DBGTOP power domain

The DBGTOP power domain is shared between all systems in SSE-700 subsystem.

Both the Secure Enclave and Host System Base System Control registers contain the BSYS_PWR_REQ.DBG_PWR_REQ field, which allows software to request the minimal power mode of the DBGTOP power domain. The DBGTOP domain can also be controlled using the

CDBGPWRUPREQ field of the DBGROM table. The expected usage is that software running on the SoC uses the Base System Control registers, while a JTAG or SWD debugger uses the CoreSight ROM tables.

Similarly to the SYSTOP domain, hardware autonomously transitions the DBGTOP domain using the values in the BSYS_PWR_REQ.DBGTOP_PWR_REQ fields, and hardware indicators, to select the lowest power mode which satisfies all the request.

The DBGTOP domain software or debug agent must do the following:

- Before attempting to access any component in the DBGTOP domain, it must either request the DBGTOP domain is powered using either the BSYS_PWR_REQ.DBGTOP_PWR_REQ of the Base System Control registers, or the CDBGPWRUPREQ field of the DBGROM table. Failure to do so will lead to the transaction completing with an error, except for accesses to the CoreSight STM-500 Extend Stimulus Port, which are treated as RAZ/WI and do not generate an error.
- After setting the BSYS_PWR_REQ.DBGTOP_PWR_REQ field to `0b1`, software must wait for BSYS_PWR_ST.DBGTOP_PWR_ST field to read as `0b1`, indicating that the DBGTOP domain is ON, before accessing the components of the DBGTOP domain.

  ——— **Note** ———

  The BSYS_PWR_REQ.DBG_PWR_REQ and BSYS_PWR_ST.DBGTOP form a handshake between software and the hardware.

  ———————————

- After setting the CDBGPWRUPREQ field of the DBGROM table to `0b1`, the debug agent must wait for the CDBGPWRUPACK field to read as `0b1`, indicating that the DBGTOP domain is ON, before accessing the components of the DBGTOP domain.

  ——— **Note** ———

  The CDBGPWRUPREQ and CDBGPWRUPRACK fields form a handshake between the debug agent and the hardware.

  Arm recommends in both cases that the debug agent only accesses the DBGTOP domain when both fields which form the handshake read as `0b1`.

  ———————————

- When either software or the debug agent no longer require the debug components, and needs to enter the DBGTOP domain into OFF, they must perform the following before:
  - Disable any trace sources they were using, (if any), and perform a flush of the trace infrastructure at the trace sink (TPIU, Host ETR or SoC ETR).
  - Disable any enabled triggers.

Arm recommends the following for controlling the DBGTOP power domain:

- Software does not change the default PPU_PWPR policy register for the DBGTOP PPU.
- Software uses the BSYS_PWR_REQ registers in the Host or Secure Enclave Base System Control blocks to control the power mode of the domain.
- JTAG or SWD debug agent uses the CDBGPWRUPREQ and CDBGPWRUPACK fields of the DBGTOP ROM table to control the power mode of the domain.
- The BSYS_PWR_REQ register of the Host System Base System Control registers is only accessed by one agent, either secure software executing on the Host CPU or software executing on the Secure Enclave. Any request to make changes to the power modes of the External System are made through messages sent to either the Host CPU or Secure Enclave using the MHUs in the system.
- When the software or the debug agent has finished using the debug, it makes sure that no the configuration of the debug components exposes any information about the system.

——— **Note** ———

This section is written with the expectation that the DBGTOP PPU power policy is set to dynamic OFF. Programming of another policy may lead to different behavior.

———————————

12.2 Time domains

In SSE-700 subsystem there are two time domains: **REFCLK** and S32K.

The **REFCLK** is used by the Host CPU, **REFCLK** timers {0-3}, Secure and Non-secure watchdogs, and can also be used by other components added by the integrator.

In the BSYS.SLEEP1 power state, the **REFCLK** time domain no longer increments and it is the responsibility of the software to restore the **REFCLK** time domain from the S32K domain, which has continued to increment in the BSYS.SLEEP1 power state.

Note

On exit from the BSYS.OFF power state, none of the two time domains are valid and must be restored from another source.

The following equation describes the relationship between the S32K time domain and the **REFCLK** time domain. The current **REFCLK** time can be calculated from the combination of the following:

- Current S32K time
- Ratio between the frequency of **S32KCLK** and **REFCLK**
- Known time values of the S32K and the **REFCLK** counters at a single point

$$t_{REF} = \left(\left(\frac{f_{REF}}{f_{32K}} \right) (t_{32K} - T_{32K}) \right) + T_{REF}$$

Figure 12-1 The relationship between S32KCLK and REFCLK

Where:

- t_{REF}: current time value in the **REFCLK** time domain
- t_{32K}: current time value in the S32K time domain
- T_{REF}: time value in the **REFCLK** time domain or CoreSight timestamp time domain at a synchronization point in the past
- T_{32K}: time value in the S32K time domain at a synchronization point in the past
- f_{REF}: frequency of **REFCLK**, in Hertz
- f_{32K}: frequency of **S32KCLK**, in Hertz

This relationship enables **REFCLK** time to be linearly scaled and offset from S32K time. The scaling factor is constant. The offse, T_{REF} and T_{32K}, must be recalculated at each power down so that all applications and software updates are preserved, as the following figure shows.

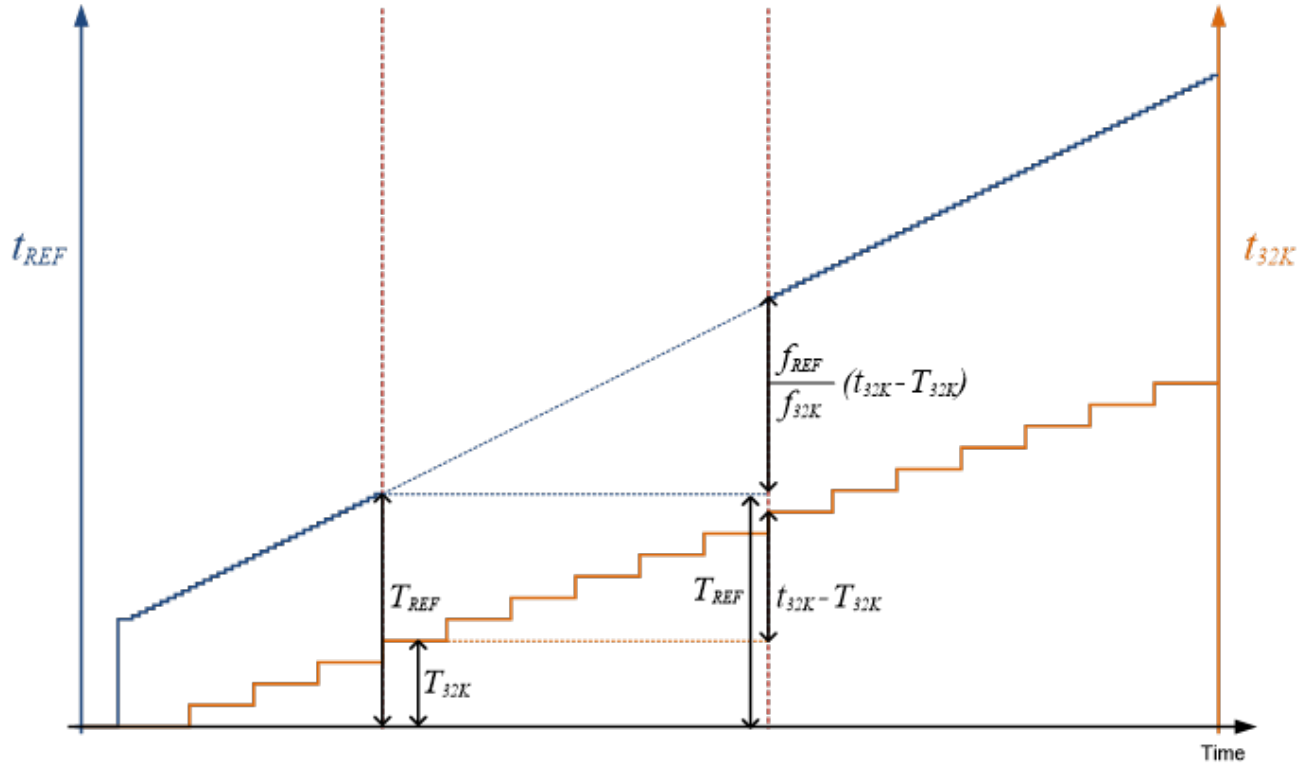

Figure 12-2 Time domain relationship

12.2.1 Restoring the REFCLK time domain

To restore the **REFCLK** time domain after it has been disabled, the software must calculate the new **REFCLK** domain time using the current S32K domain time and update the value of the **REFCLK** Counter.

For maximum accuracy this update can be timed to a **S32KCLK** clock edge. *11.3.4 REFCLK Counter CNTControl register summary* on page 11-216 describes the registers that enable the **REFCLK** time value to be set against the rising edge of **S32KCLK**.

12.3 Debug agents

This section describes the SSE-700 debug agents.

This section contains the following subsections:

12.3.1 Certificate injection

This section describes the procedure to inject a certificate to enable SSE-700 debug privileges.

The SSE-700 subsystem has a lifecycle state, which is used to define default values for CoreSight Authentication signals used within the SoC.

Note

The lifecycle state of the SoC is managed by the Secure Enclave of SSE-700 subsystem. The lifecycle state can be one of the following:

- Chip Manufacture
- Device Manufacture
- Secure Enable
- Return Merchandise Authorization

For more information on the lifecycle states of the SoC, see the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

For example, in the Secure Enable lifecycle state all CoreSight Authentication signals are LOW indicating that all debug functionality is disabled. To enable a JTAG/SWD debugger to access the debug infrastructure of the SSE-700 subsystem and wider SoC, the SSE-700 subsystem includes a CoreSight SDC-600. The SDC-600 provides a standard method for a JTAG debugger to communicate with software running on the SoC.

Note

The CoreSight SDC-600 is accessible by both JTAG/SWD and the software running on the SoC. Arm recommends that software does not use the CoreSight SDC-600 for certificate injection and instead communicates directly with the software controlling the CoreSight Authentication signals. For software outside the Secure Enclave this will require the use of the MHUs to the Secure Enclave.

The CoreSight SDC-600 is split into two components EXT APBCOM and INT APBCOM. For more information on these components, see the *Arm® CoreSight™ SDC-600 Secure Debug Channel Technical Reference Manual*. The EXT APBCOM is used by the JTAG/SWD debug agent and the INT APBCOM is used by software in the SSE-700 subsystem. The INT APBCOM is part of the Host System memory map but the interrupt can be routed to either the Host CPU or the Secure Enclave, through the Interrupt Router. It is IMPLEMENTATION DEFINED, whether software running on the Host CPU or the Secure Enclave manages the SDC-600 INT APBCOM.

An example sequence for injecting a certificate using a JTAG/SWD debugger is as follows:

1. Debugger connects to the target SoC and reads the DP ROM table to discover the location of the EXT APBCOM.
2. Debugger uses the EXT APBCOM to establish a link with the software running on the target SoC, using the steps defined in *Advanced Communications Channel Architecture Specification*. The software running on the target SoC uses the INT APBCOM to receive messages from the debug agent.
3. When the link is established, the debugger transmits the certificate. Software receives the certificate and validates it and does one of the following:

- If the certificate is valid, it updates the required Security Control Bits, which drive the associated CoreSight Authentication signals for the DAZ and DAACGs. It also provides an acknowledgment back to debugger indicating successful validation.
- If the certificate is invalid, no update to the Security Control Bits occurs. It is IMPLEMENTATION DEFINED whether an error message is provided back to the debugger and what action the system and debugger takes when this occurs.

4. Debugger either:
   - If the requested debug privileges have been enabled, the debugger continues with the debug activity.
   - If the requested debug privileges were not enabled, the debugger retries or reports an error.

When the debugger has finished sending messages through SDC-600, it must terminate the link using the steps defined in the *Advanced Communications Channel Architecture Specification*.

Arm recommends that before transitioning the device to a lifecycle state where debug functionality is disabled, the code used to perform certificate authentication has been tested.

SSE-700 only supports the following ways of injecting a certificate:

Runtime authentication
: The certificate is inserted at any time and processed by the target system.

Boot-time authentication, where nSRST is used to enter boot mode
: The certificate is inserted at boot time. The debug causes the SoC to perform a boot by causing an **nSRST** reset to occur. Triggering of **nSRST** can be done using either the **nSRST** or setting CSYSRSTREQ in the DP ROM to `0b1`.

12.3.2 Debug from Reset

Debug from Reset is the process of debugging the following CPUs from reset release in the SSE-700 subsystem:

- Secure Enclave Cortex-M0+
- Host CPU
- External System's CPUs

To perform Debug from Reset on the Secure Enclave Cortex-M0+ the debug agent using the JTAG/SWD must follow the following steps:

1. Debug agent either asserts **nSRST** input or sets the CSYSRSTREQ bit in the DP ROM table and wait for the CSYSRSTACK bit to become `0b1`.
2. Debug agent configures the debug functionality required. It might be required that the debug agent perform power requests to be able to access certain debug registers. For example, the debug logic of the Host CPU core.
3. Once the debug agent has finished configuring the required debug functionality, it either de-asserts the **nSRST** input or clears the CSYSRSTREQ bit in the DP ROM table. At this point the boot process continues as normal.

It is possible for the CPU of the Host and External System to perform Debug from Reset. The method by which software running on SSE-700 enables Debug from Reset of the Host and External Systems is IMPLEMENTATION DEFINED.

Whether it is possible to perform Debug from Reset, depends on the settings of the DAZs in the SSE-700 subsystem. By default, in the Secure lifecycle the SECENCAUTH prevents debug access to the Secure Enclave Cortex-M0+ and the debug agent is required to insert a certificate to request access. The processing of a certificate requires software to run on the SSE-700. It is not possible to insert the certificate and then cause a debug reset using either the **nSRST**, or DP ROM **CSYSRSTREQ/ACK** handshake as this resets the SCBs to the default value for the lifecycle state.

Arm strongly recommends the following:

- The Secure Enclave ROM code includes the ability to handle certificate insertion using the SDC-600 to allow for debugging of all non-ROM based boot code.
- The Secure Enclave Cortex-M0+ checks for a possible certificate insertion as part of the boot process. This allows a debug agent to start inserting a certificate whilst preventing the Secure Enclave Cortex-M0+ from executing software using either the **nSRST** input or the DP ROM CSYSRSTREQ/ACK handshake.
- Before transitioning to the Secure lifecycle state, the ability of the Secure Enclave Cortex-M0+ to process a certificate is confirmed working, to allow for debug as early as possible in the boot process.

It might be required to de-assert **nSRST** input or clear CSYSRSTREQ field in the DP ROM earlier if the software is required to perform certain actions. For example, powering a specific domain. If this is required, then an IMPLEMENTATION DEFINED software mechanism can be implemented to indicate that a debug agent is performing Debug from Reset on a specific target processor. If the debug agent is performing Debug from Reset on more than one processor in the SoC, and one of these processors is required to be released from reset to enable access, the debug agent must stage the configuration of the debug. At first, it programs the debug logic of the processor, which must be released from reset before de-asserting **nSRST** input or clearing CSYSRSTREQ field in the DP ROM. Once the other processor is accessible, it continues with the configuration of the debug.

A debugger can also use the **nSRST** input or DP ROM CSYSRSTREQ bit to reset the SoC to a known state without the need for Debug from Reset. In this case, the debug agent follows the steps above but does not perform any configuration at step 2 of the process.

12.3.3 Using the External Debug Bus

The SSE-700 subsystem has an External Debug Bus enabling access to the Access Ports included in the SoC from the Host System, Secure Enclave or an External System.

The debug agent must not do the following:

- Use an Access Port to target a location on the External Debug Bus. For example, using the AXI-AP to access the Access Port of an External System. The debug agent should instead directly access the Access Port of the External System.
- Use the External Debug Bus to perform self-hosted debug of the system on which the debug agent is running. For example, to perform self-hosted debug of the Host system, the debug agent running on the Host CPU should access the debug components using the Host System Debug region of the Host System memory map.

Arm expects the External Debug Bus to be used in one of the following scenarios:

- Debug agent running on the Host CPU to debug one or more of the External Systems
- Debug agent running on an External Systems to debug one or more of the other External Systems and/or the Host System

Note

In both scenarios, the system on which the Debug agent is running can also be being debugged by the debug agent, however, the debug agent must not use the External Debug Bus to configure the debug components of its system or to perform any memory access to the system's memory map.

It is possible to have the debug agent run on the Secure Enclave. But this is not the expected use-case due to the fact that the Secure Enclave is providing the Root of Trust to the system. Arm strongly recommends that the Secure Enclave is not used to run the debug agent.

It is possible for the Secure Enclave to be included in the list of systems being debugged in the scenarios above. However, this is not expected as the debug agent will typically be running in an environment that is less trusted compared to the Secure Enclave. Allowing it to debug the Secure Enclave would compromise security. Therefore, Arm strongly recommends that the Secure Enclave is only debugged by a debug agent connecting through the Debug Port.

12.4 Host and External System {0-1} reset request

For both the Host and External System {0-1} reset requests, there is a handshake between software and the Reset Controller using one of the following:

- HOST_SYS_RST_CTRL and HOST_SYS_RST_ST registers for the Host System reset request.
- EXT_SYS{0-1}_RST_CTRL and EXT_SYS{0-1}_RST_ST registers for the External System {0-1} reset request.

Software must follow this sequence:

1. Set RST_REQ to `0b1`.
2. Poll RST_ACK until the value is either `0b01` or `0b10`.
3. If the value is `0b10`, the requested reset has been accepted:
   a. Set RST_REQ to `0b0`.
   b. Wait for RST_ACK to become `0b00`.
4. If the value is `0b01`, the requested reset has been denied:
   a. Set RST_REQ to `0b0`.
   b. Wait for RST_ACK to become `0b00`.
   c. Software can then do one of the following:
      - Repeat the sequence again.
      - Request a large reset of the SoC:
        — For the External System {0-1} reset request, software can request a Host System or SoC reset
        — For the Host System reset request, software can request an SoC reset

In the sequence, above depending on whether software is attempting to reset the Host or an External System, the RST_REQ and RST_ACK fields are located in the:

- HOST_SYS_RST_CTRL and HOST_SYS_RST_ST respectively for the Host System reset request
- EXT_SYS{0-1}_RST_CTRL and EXT_SYS{0-1}_RST_ST respectively for the External System {0-1} reset request

If software sets RST_REQ back to `0b0` before RST_ACK becomes `0b01` or `0b10`, software cannot guarantee whether the reset request was applied or not.

Note

Arm recommends that software does not repeatedly request a reset of the system if it gets denied and requests a wider reset of the SoC instead.

To guarantee that the reset request is completed successfully, Arm recommends:

- To reset an External System the External System is in a quiescent state with no outstanding transactions from or to the External System.
- To reset the Host System there are no outstanding accesses from the Secure Enclave into the Host System address space.

12.5 Watchdog usage

SSE-700 subsystem implements four different watchdogs. Each one is used for a specific functionality:

Non-secure watchdog

The Non-secure watchdog is used by the Rich OS to guard against it becoming unresponsive. The Non-secure watchdog uses the REFCLK time domain and is not operational in the BSYS.SLEEP1 power state. On the first expiry of the watchdog an interrupt is generated to the Host CPU and is taken by the Rich OS. If the watchdog expires for a second time, a second interrupt is generated. The second interrupt is taken by the Host CPU Secure firmware which takes actions to handle the unresponsive Rich OS.

Secure watchdog

The Secure watchdog is used by the Host CPU Secure firmware to guard against it becoming unresponsive. The Secure watchdog uses the **REFCLK** time domain and is not operational in the BSYS.SLEEP1 power state. On the first expiry of the watchdog an interrupt is generated to the Host CPU and is taken by the Host CPU Secure firmware. If the watchdog expires for a second time, a second interrupt is generated and routed to the Secure Enclave. The Secure Enclave takes actions to handle the unresponsive Host CPU Secure firmware.

Secure Enclave watchdog

The Secure Enclave watchdog guards against the Secure Enclave becoming unresponsive. The Secure Enclave watchdog uses the **SECENCDIVCLK** and is not operational when the SECENCTOP power domain enters the OFF or MEM_RET power-modes. On first expiry of the watchdog, an interrupt is generated to the Secure Enclave Cortex-M0+. If the watchdog expires for a second time a reset request is generated and causes a reset of the entire SoC.

SoC Watchdog

The SoC watchdog guards against the entire SoC becoming unresponsive. It is operational in all power states of SSE-700 and uses the **S32KCLK**. On first expiry of the watchdog an interrupt is generated to the Secure Enclave Cortex-M0+. If the watchdog expires for a second time a reset request is generated and causes a reset of the entire SoC.

The SoC watchdog uses **S32KCLK** to decrement the watchdog, and is allowed to be enabled when the **S32KCLK** is not stable. This is different to the S32K time domain that can only be enabled once the **S32KCLK** is stable. However, Arm recommends that software programs a value which is long enough so not to cause unexpected watchdog reset requests to be generated.

——— **Note** ———

For more information on the Secure Enclave watchdog and SoC watchdog, see the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

———————————

All watchdogs must be enabled before the watchdog generates any interrupt or reset request.

Arm recommends the following:

- Software configures the debug infrastructure and watchdogs to halt when performing halted debug.
- Software disables the Non-secure or Secure watchdogs when entering the BSYS.SLEEP1 power state. This avoids a watchdog expiry occurring whilst returning to the BSYS.SLEEP0 or BSYS.RUN power states.
- Secure Enclave firmware disables the Secure Enclave watchdog before entering the SECENCTOP power domain into the OFF or MEM_RET power-modes.

12.6 Lifecycle

An SoC based on SSE-700 subsystem has a lifecycle state as defined in *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

The Lifecycle state is managed by an IMPLEMENTATION DEFINED method and the method for advancing the lifecycle state is outside the scope of SSE-700 subsystem, however, Arm makes the following recommendations:

- Before advancing the lifecycle to a state whereby debug functionality is disabled by default, there must be a method for a debug agent to request that the debug functionality be re-enabled, using the method described in *12.3.1 Certificate injection* on page 12-244.
- Before advancing the lifecycle to a state where the Bypass interface of the Host System Firewall is driven to 0 by default, the Secure Enclave firmware has a method to enable the Bypass of the Firewall. To enable the Bypass on the Host System Firewall, the Secure Enclave firmware must update the SCB to drive bit [37] to `0b1`.

  ——— **Note** ———

  Arm recommends that this is only included as part of a debug certificate, for any details of SCB see the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*

  ———————————

- Before advancing the lifecycle to a state whereby the Bypass interface of the Secure Enclave Firewall is driven to 0 by default, the Secure Enclave firmware is able to program the Firewall to an extent where Host CPU is able to boot.

12.7 Lock control

The SSE-700 provides several locks which software can use to prevent update to certain components and configuration registers.

——— **Note** ———

The value read in HOST_SYS_LCTRL_ST register can be a stale copy and software must handle this. The software sequences below take this into consideration.

———————————

Host CPU lock

The Host CPU lock is controlled using the HOST_CPU{0-3}_LOCK field in the HOST_SYS_LCTRL_{SET/CLR} registers. The status of the lock can be seen in the HOST_SYS_LCTRL_ST.HOST_CPU{0-3}_LOCK field.

The Host CPU lock is cleared automatically by hardware when:

- The Core{0-3} power domain enters one of the following power modes: OFF, OFF_EMU, or WARM_RST
- The CLUSTOP power domain enters one of the following power modes: OFF, MEM_RET, or WARM_RST

The software sequence to set the Host CPU lock is as follows:

1. Core enters the ON power-mode.
2. Software performs configuration of the registers which are affected by the **CP15SDISABLE** signal. For more information, see *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. Software can perform other steps before this, but Arm recommends this is kept to a minimum.
3. Software writes `0b1` to the HOST_SYS_LCTRL_SET.HOST_CPU{0-3}_LOCK field associated with the core.
4. Software checks the status of the lock by reading the HOST_SYS_LCTRL_ST.HOST_CPU{0-3}_LOCK field.

——— **Note** ———

Arm strongly recommends the software performing this sequence is executing on the Host CPU core to be locked.

If the software performing the sequence is not executing on the Host CPU core to be locked, Arm strongly recommends that the Host CPU core to be locked does not enter any of the following power modes: OFF, OFF_EMU, or WARM_RST once it has completed step 2 until it has performed step 4. This avoids race conditions between clearing and setting the lock.

———————————

The software sequence to clear the Host CPU lock is as follows:

1. Software writes `0b1` to HOST_SYS_LCTRL_CLR.HOST_CPU{0-3}_LOCK field associated with the core to be unlocked.
2. Software waits for the associated HOST_CPU{0-3}LOCK field in the HOST_SYS_LCTRL_ST register to become `0b0`, before performing any configuration of the locked registers in the Host CPU core.

Host GIC lock

The Host GIC lock is controlled using the HOST_GIC_LOCK field in the HOST_SYS_LCTRL_{SET/CLR} registers. The status of the lock can be seen in the HOST_SYS_LCTRL_ST.HOST_GIC_LOCK field.

The Host GIC lock is cleared automatically by hardware when the CLUSTOP power domain enters one of the following power modes: OFF, MEM_RET or WARM_RST.

The software sequence to set the Host GIC lock is as follows:

1. Configuration of the GIC configuration registers affected by the **CFGSDISABLE** signal. For more information, see *Arm® Generic Interrupt Controller Architecture Specification, architecture version 2.0*.
2. Software writes `0b1` to the HOST_SYS_LCTRL_SET.HOST_GIC_LOCK field.
3. Software checks the status of the lock by reading the HOST_SYS_LCTRL_ST.HOST_GIC_LOCK. If the field is `0b0` then software repeats the sequence.

Note

Arm strongly recommends that software performing this sequence is excuting on one of the Host CPU cores.

If the software performing this sequence is not executing on one of the Host CPU cores, Arm strongly recommends that the CLUSTOP domain does not enter any of the following power modes: OFF, MEM_RET or WARM_RST during the sequene. This avoids race conditions between clearing and setting the lock.

The software sequence to clear the HOST CPU lock is as follows:

1. Software writes `0b1` to HOST_SYS_LCTRL_CLR.HOST_GIC_LOCK field.
2. Software waits for the HOST_SYS_LCTRL_ST.HOST_GIC_LOCK field to become `0b0`, before performing any configuration of the locked registers in the Host GIC.

Other locks

The SSE-700 provides a Host Base System Control, Interrupt Router and Host System Firewall lock. The locks are controlled using the HOST_LOCK, INT_RTR_LOCK and HOST_FW_LOCK fields respectively, in the HOST_SYS_LCTRL_{SET/CLR} registers. The status of the lock can be seen in the same field in the HOST_SYS_LCTRL_ST register.

Note

For the Interrupt Router and Host System Firewall lock, the status of the lock can also be seen using registers in the component.

Lock Clear Disable

The default behavior of the locks in the SSE-700 allows software to remove the lock by writing `0b1` to the associated field in the HOST_SYS_LCTRL_CLR register. This behavior can be modified by setting the HOST_SYS_LCTRL_ST.LOCK_CLR_DIS field to `0b1`. This is done by software writing `0b1` to HOST_SYS_LCTRL_SET.LOCK_CLR_DIS field.

When the HOST_SYS_LCTRL_ST.LOCK_CLR_DIS field is `0b1` any writes to the HOST_SYS_LCTRL_CLR register are ignored and software cannot clear any locks.

Note

When the HOST_SYS_LCTRL_ST.LOCK_CLR_DIS is `0b1`, the HOST_SYS_LCTRL_ST.{HOST_CPU{0-3}_LOCK/HOST_GIC_LOCK} fields are still set to `0b0` when any of the conditions stated in sections *Host CPU lock* on page 12-250 and *Host GIC lock* on page 12-250 occur.

Appendix A
Message Handling Unit

This appendix describes the *Message Handling Unit* (MHU) component of the SSE-700.

It contains the following sections:

A.1 Overview

In a SoC there can be multiple systems, each with their own processing elements. Each system executes a different software stack. To perform the overall functionality of the SoC, the different software entities need to communicate.

The following systems are some of the many systems in an example SoC for a mobile phone:

- Application Processor: Executes the RichOS, such as Linux
- Communication Processor: A modem or WIFI system, which provides communication to an external network
- System Control Processor: System control functionality, such as power and clock control
- Secure Enclave: A secure system which provides security functionality to the rest of the SoC. To reduce the possibility of a malicious agent gaining access to secrets, this is typically implemented as a separate system.

For the SoC to function as designed, the different software entities operating on each of these systems need to communicate with one another. There are different communication methods, for example:

- System calls, such as SVC or SMC
- Events
- Interrupts

Each type of communication has advantages and disadvantages and is suitable in different situations.

The usage of system calls and events for communication is considered outside the scope of this chapter. For the remainder of this chapter, only interrupt-based communication is considered.

This section contains the following subsection:

- *A.1.1 Interrupt Based Communication* on page Appx-A-253.

A.1.1 Interrupt Based Communication

At the lowest level, interrupt communication depends on a Sender passing an indication of an event to a Receiver. This event indication is generated by a software operation, that is, an instruction or memory write.

This event indication is often passed using a physical signal called an interrupt to the Receiver. Alongside the interrupt, additional information can be included either in-band or out-band. The interrupt and additional information form a transfer, sent by the Sender to the Receiver. A number of transfers then form a complete message, with the number of messages dependent on the message protocol.

A.2 Message Handling Unit v2

An MHU is made up of two memory mapped register frames.

The first frame is used by the sender of the transfer, referred to as the Sender. The second frame is used by the receiver of the transfer, referred to as the Receiver. Inside each frame, there are:

- Several Channels which the Sender uses to send transfers to the Receiver
- Control and identification registers

This section contains the following subsections:

A.2.1 Channel

An MHU has several Channels, each of which has the following features:

- Channel Status (CH_ST) register: Shows the status of the Channel
- Channel Status Mask (CH_ST_MSK) register: Shows the status of the Channel with the Channel Mask applied
- Channel Set (CH_SET) register: Sets bits in the Channel Status and Status Mask
- Channel Clear (CH_CLR) register: Clears bits in the Channel Status and Status Mask
- Channel Mask Status (CH_MSK_ST) register: Shows the status of the Channel Mask, used with the Channel Status register, to generate the Channel Status Mask
- Channel Mask Set (CH_MSK_SET) register: Sets bits in the Channel Mask
- Channel Mask Clear (CH_MSK_CLR) register: Clears bits in the Channel Mask
- Channel Interrupt Status (CH_INT_ST) register: Shows the status of the channel's clear interrupt
- Channel Interrupt Clear (CH_INT_CLR) register: Clears the channel's clear interrupt
- Channel Interrupt Enable (CH_INT_EN) register: Enable for the channel's clear interrupt
- Interrupt: Indicates when a transfer has occurred to the Receiver

The interrupt of a Channel is generated when any bit in the CH_ST_MSK register is set to 1. The interrupt is level-based and remains asserted until all bits in the register are set to 0.

A Channel can be subdivided into 32 flags where each bit of the register controls a different flag or can be used as transfer payload register. How the Channel is used depends upon the transport protocol used. See *A.3 Transport protocols* on page Appx-A-270 for more information.

A.2.2 Transfers

Transfers in the MHUv2 are sent between the Sender and Receiver. At its most basic form a transfer is sent by:

- The Sender writing `0b1` to one or more of the flag bits in the CH_SET register.
- This is reflected in:
  - The CH_ST registers of both the Sender and Receiver
  - The CH_ST_MSK register, of the Receiver, depending on the current values of CH_MSK_ST.
- If any bit in the CH_ST_MSK is `0b1`, then the interrupt for the Channel is asserted.
- The Receiver of the transfer, must clear the transfer by writing `0b1` to one or more flag bits in the CH_CLR register.

How Sender and Receiver use the individual Channels and flags bits depends on the transport protocol used. See *A.3 Transport protocols* on page Appx-A-270 for more information on the supported transport protocols.

The number of transfers used to create a message is dependent on:

- Size of the message being sent
- Transport protocol that is used
- Number of Channels that are implemented in the MHU
- Number of concurrent transfers between the Sender and Receiver.

A.2.3 Ready to Send

The MHUv2 is designed to allow two entities to communicate. It is not guaranteed that when the Sender attempts to send a transfer, the Receiver can receive it.

To enable the Sender to request that the Receiver enters a state where it can receiver a transfer, the Sender uses the ACCESS_REQUEST and ACCESS_READY registers. The ACCESS_REQUEST register enables the Sender to request that the Receiver enters a state to be able to receiver a transfer. The ACCESS_READY indicates whether the Receiver is in a state to Receiver a transfer.

When **ACCESS_READY.ACC_RDY** is LOW, any attempt by the Sender to access any registers of the Channel Windows is treated as follows:

- RAZ/WI with no error generated when **RESP_CFG.NR_RDY** is `0b0`
- RAZ/WI with an error generated when **RESP_CFG.NR_RDY** is `0b1`

Software requirements

Even when setting **ACCESS_REQUEST.ACC_REQ** to `0b1`, software is still responsible for guaranteeing that the transfer has been received by the Receiver using an IMPLEMENTATION DEFINED software protocol. An example of a sequence is:

1. Sender configures the MHU to enable the NR2R and R2NR interrupts, by setting the INT_EN.NR2R and INT_EN.R2NR fields to `0b1`.
2. Sender sets the **ACCESS_REQUEST.ACC_REQ** field to `0b1`.
3. Sender reads the status of the **ACCESS_REQUEST.ACC_RDY** field. If it is `0b1` then it continues to step 4. Otherwise, the Sender waits for the NR2R interrupt to be triggered.

   ——— **Note** ———

   A NR2R interrupt is generated due to the Sender setting the **ACCESS_REQUEST.ACC_REQ** field to `0b1`, if **ACCESS_READY.RDY** was already 1. Software must clear this interrupt before continuing.

   ———————————

4. Sender, using one of the transport protocols defined in *A.3 Transport protocols* on page Appx-A-270, sends the transfer to the Receiver. While the Sender is performing the steps of the transport protocol, the transfer is considered to be outstanding and is not guaranteed to reach the Receiver. While performing the transport protocol, the Sender may receive either:

   - A R2NR interrupt
   - An error response to an access to a Channel window register if RESP_CFG.NR_RDY is set to `0b1`

   This indicates that any transfers which were outstanding may have been lost. The Sender must take the following action:

   a. Clear the R2NR interrupt and process the fault handler, if an error was generated.
   b. Perform an IMPLEMENTATION DEFINED recovery sequence. This sequence can be to resend the transfer when the Receiver returns to a state where it is able to receive transfers or to send other transfers to resynchronize the Sender and Receiver.

——— **Note** ———

When the Receiver enters a state where it cannot receive a transfer, the Receiver may have been reset. Therefore, both the Sender and Receiver may need to be resynchronized before continuing to communicate.

———————————

5. Sender sets the **ACCESS_REQUEST.ACC_REQ** field to `0b0` if it has no further transfer to send. If the Sender has more transfers to send, it repeats step 4 until it has no more transfers to send.

The Sender of the transfer may also enter a state where it cannot complete the sending of the transfer. For example, the Sender may be reset by a watchdog while writing to the CH_SET register. It is the responsibility of the Sender and Receiver to clear and ignore any transfers which were from before the Sender was reset.

——— **Note** ———

When the Sender enters a state where it cannot complete sending a transfer, the Sender may have been reset. Therefore, both the Sender and Receiver may need to resynchronize before continuing to communicate.

———————————

A.2.4 Interrupts

The Sender and Receiver frames both have interrupts.

All interrupts in the MHU:

- Are level-based
- Have a register for the status, to clear and enable the interrupt

The Sender frame has three interrupts:

- Not Ready to Ready interrupt (NR2R):
  - This interrupt is asserted when INT_EN.NR2R and INT_ST.NR2R are `0b1`.
  - INT_ST.NR2R becomes `0b1`, when:
    - ACC_RDY goes high and ACC_REQ is `0b1`.
    - ACC_REQ goes high and ACC_RDY is `0b1`.
  - INT_ST.NR2R becomes `0b0`, when software writes `0b1` to INT_CLR.NR2R in the Sender's frame.
- Ready to Not Ready (R2NR):
  - This interrupt is asserted when INT_EN.R2NR and INT_ST.R2NR are `0b1`.
  - INT_ST.R2NR becomes `0b1`, when ACC_RDY goes LOW.
  - INT_ST.R2NR becomes `0b0`, when software writes `0b1` to INT_CLR.R2NR in the Sender's frame.
- Combined Interrupt (CHCOMB):
  - This interrupt is asserted when the Sender frame's INT_EN.CHCOMB is `0b1` and any of the following occur:
    - Any Channel's CH_INT_ST.CH_CLR and CH_INT_EN.CH_CLR are both `0b1`.
    - INT_ST.NR2R and INT_EN.NR2R are both `0b1`.
    - INT_ST.R2NR and INT_EN.R2NR are both `0b1`.

The CH_CLR bits relate to the Channel Clear interrupt. This removes the need for software to poll the status register, to detect when the Receiver has finished processing the previous transfer. The clear interrupt, for the Channel, is generated when the Receiver writes to the CH_CLR register of the Channel, irrespective of the value written.

Note

Sender is still required to check the value of the CH_ST register, as defined by the transport protocol, when it receivers the clear interrupt.

The clear interrupt is only required to be received by the Sender of the original message if the Ready to Send interface remains in the state where both signals are `0b1`. This means that software must not set the ACCESS_REQUEST.ACC_REQ bit to `0b0` until it has received the clear interrupt.

Note

It is allowed for the clear interrupt to be generated when the Ready to Send interface is not in the state where both signals are `0b1`.

The Receiver frame has a combined interrupt output that is set when the Receiver frame's INT_EN.CHCOMB is `0b1` and at least one bit in any channel's CH_ST_MSK register is `0b1`. To clear the combined interrupt, all channel CH_ST_MSK bits must be cleared.

A.2.5 Programmers Model

The MHU only supports 32-bit word aligned read/write access. Any access attempt that does not meet this requirement will be treated as an aligned 32-bit access.

Sender Frame registers

This section provides details of the registers of the Sender Frame of the MHU.

Table A-1 Sender Frame register

Offset	Access type	Register name	Short name
`0x000` – `0xF7C`	-	Sender Channel Window 0 -123	-
`0xF80`	RO	Message Handling Unit Configuration	MHU_CFG
`0xF84`	RW	Response Configuration	RESP_CFG
`0xF88`	RW	Access Request	ACCESS_REQUEST
`0xF8C`	RO	Access Ready	ACCESS_READY
`0xF90`	RO	Interrupt Status	INT_ST
`0xF94`	WO	Interrupt Clear	INT_CLR
`0xF98`	RW	Interrupt Enable	INT_EN
`0xF9C`	RO	Reserved	-
`0xFA0`	RO	Channel interrupt status for channels 0 – 31	CHCOMB_INT_ST0
`0xFA4`	RO	Channel interrupt status for channels 32 – 63	CHCOMB_INT_ST1
`0xFA8`	RO	Channel interrupt status for channels 64 - 95	CHCOMB_INT_ST2
`0xFAC`	RO	Channel interrupt status for channels 96 – 123	CHCOMB_INT_ST3
`0xFB0` – `0xFC4`	RO	Reserved	-
`0xFC8`	RO	Implementer Identification Register	IIDR
`0xFCC`	RO	Architecture Identification Register	AIDR
`0xFD0`	RO	Peripheral ID4	PID4
`0xFD4`	RO	Peripheral ID5	PID5
`0xFD8`	RO	Peripheral ID6	PID6
`0xFDC`	RO	Peripheral ID7	PID7

Table A-1 Sender Frame register (continued)

Offset	Access type	Register name	Short name
0xFE0	RO	Peripheral ID0	PID0
0xFE4	RO	Peripheral ID1	PID1
0xFE8	RO	Peripheral ID2	PID2
0xFEC	RO	Peripheral ID3	PID3
0xFF0	RO	Component ID0	CID0
0xFF4	RO	Component ID1	CID1
0xFF8	RO	Component ID2	CID2
0xFFC	RO	Component ID3	CID3

Receiver Frame registers

This section provides details of the registers of the Receiver Frame of the MHU.

Table A-2 Receiver Frame register

Offset	Access type	Register name	Short name
0x000 – 0xF7C	-	Receiver Channel Window 0 -123	-
0xF80	RO	Message Handling Unit Configuration	MHU_CFG
0xF84 – 0xF8C	RO	Reserved	-
0xF90	RO	Interrupt Status	INT_ST
0xF94	RO	Interrupt Clear	INT_CLR
0xF98	RW	Interrupt Enable	INT_EN
0xF9C	RO	Reserved	-
0xFA0	RO	Channel Interrupt Status for channels 0 – 31	CHCOMB_INT_ST0
0xFA4	RO	Channel Interrupt Status for channels 32 – 63	CHCOMB_INT_ST1
0xFA8	RO	Channel Interrupt Status for channels 64 – 95	CHCOMB_INT_ST2
0xFAC	RO	Channel Interrupt Status for channels 96 – 123	CHCOMB_INT_ST3
0xFB0 – 0xFC4	RO	Reserved	-
0xFC8	RO	Implementer Identification Register	IIDR
0xFCC	RO	Architecture Identification Register	AIDR
0xFD0	RO	Peripheral ID4	PID4
0xFD4	RO	Peripheral ID5	PID5
0xFD8	RO	Peripheral ID6	PID6
0xFDC	RO	Peripheral ID7	PID7
0xFE0	RO	Peripheral ID0	PID0
0xFE4	RO	Peripheral ID1	PID1
0xFE8	RO	Peripheral ID2	PID2
0xFEC	RO	Peripheral ID3	PID3
0xFF0	RO	Component ID0	CID0
0xFF4	RO	Component ID1	CID1

Table A-2 Receiver Frame register (continued)

Offset	Access type	Register name	Short name
0xFF8	RO	Component ID2	CID2
0xFFC	RO	Component ID3	CID3

Channel windows

A channel window is a group of registers. The registers in the channel window vary depending on whether it is the Sender or Receiver view.

There can be between 1 and 124 channels in an MHU. The number of channels implemented can be discovered using MHU_CFG.NUM_CH.

Each channel occupies eight 32-bit words in both the Sender and Receiver register maps.

The address space that is allocated to channels that are not implemented, is Reserved and treated as RAZ/WI.

Sender Channel Window

This section provides details of the registers of the Sender Frames view of a Channel Window.

Table A-3 Sender Channel Window register overview

Offset	Access type	Register name	Short name
0x00	RO	Channel Status	CH_ST
0x04	RO	Reserved	-
0x08	RO	Reserved	-
0x0C	WO	Channel Set	CH_SET
0x10	RO	Channel Interrupt Status	CH_INT_ST
0x14	WO	Channel Interrupt Clear	CH_INT_CLR
0x18	RW	Channel Interrupt Enable	CH_INT_EN
0x1C	RO	Reserved	-

Receiver Channel Window

This section provides details of the registers of the Receiver Frame view of a Channel Window.

Table A-4 Receiver Channel Window register overview

Offset	Access type	Register name	Short name
0x00	RO	Channel Status	CH_ST
0x04	RO	Channel Status Masked	CH_ST_MSK
0x08	WO	Channel Clear	CH_CLR
0x0C	RO	Reserved	-
0x10	RO	Channel Mask Status	CH_MSK_ST
0x14	WO	Channel Mask Set	CH_MSK_SET
0x18	WO	Channel Mask Clear	CH_MSK_CLR
0x1C	RO	Reserved	-

Channel Status (CH_ST)

This register shows the status of the channel. This register is part of both the Receiver and Sender channel windows.

Table A-5 Channel Status register description

Bit	Name	Description	Type	Reset
[31:0]	FLAG{x}	Channel flag status. Each bit can be used as individual flags or bits can be grouped. The usage depends upon the transport protocol used. Bits in this register are set by writing `0b1` to the corresponding bit in the CH_SET register. Bits in this register are cleared by writing `0b1` to the corresponding bit the CH_CLR register. If software: • Sets a bit that is already set, the bit remains set. • Clears a bit that is already cleared, the bit remains cleared. • Sets and clears a bit at the same time, the bit remains set. Arm strongly recommends that software follows the transport protocols defined in *A.3 Transport protocols* on page Appx-A-270.	RO	`0x0000_0000`

Note

If the Receiver frame is reset then the contents of the CH_ST register in the Sender frame is also reset. Software is responsible for the handling of lost messages if this occurs.

Channel Status Masked (CH_ST_MSK)

This register shows the status of the channel with the channel mask applied. This register is part of the Receiver channel window.

Table A-6 Channel Status Masked register description

Bit	Name	Description	Type	Reset
[31:0]	FLAG_MSK{x}	Channel flag status after the mask is applied. When this field is nonzero, the interrupt for the Channel is asserted. The value in this register is equal to CH_ST and ~CH_MSK_ST, at the point the read occurs.	RO	`0x0000_0000`

Channel Clear (CH_CLR)

This register clears bits in the Channel Status and Status Mask registers. This register is part of the Receiver channel window.

Table A-7 Channel Clear register description

Bit	Name	Description	Type	Reset
[31:0]	FLAG_CLR{x}	Channel flag clear. Write `0b1` to a bit clears the corresponding bit in the CH_ST and CH_ST_MSK. Writing `0b0` has no effect. Each bit always reads as `0b0`.	WO	`0x0000_0000`

Channel Set (CH_SET)

This register sets bits in the Channel Status and Channel Status Mask registers. This reguster is part of the Sender channel window.

Table A-8 Channel Set register description

Bit	Name	Description	Type	Reset
[31:0]	FLAG_SET{x}	Channel flag set. Write `0b1` to a bit sets the corresponding bit in the CH_ST. Writing `0b0` has no effect. Each bit always reads as `0b0`.	WO	`0x0000_0000`

Channel Mask Status (CH_MSK_ST)

This register shows the status of the channel mask, used with the Channel Status register, to generate the channel status mask. This register is part of the Receiver channel window.

Table A-9 Channel Mask Status register description

Bit	Name	Description	Type	Reset
[31:0]	FLAG_MSK{x}	Channel flag mask. `0b0` – Flag bit is unmasked. When a bit is unmasked, the corresponding bits in the CH_ST and CH_ST_MSK registers have the same value. `0b1` – Flag bit is masked. When a bit is masked, the corresponding bit in the CH_ST_MSK register always reads as `0b0`.	RO	`0x0000_0000`

Channel Mask Set (CH_MSK_SET)

This register sets bits in the channel mask. This register is part of the Receiver channel window.

Table A-10 Channel Mask Set register description

Bit	Name	Description	Type	Reset
[31:0]	FLAG_MSK_SET{x}	Channel flag mask set. Write `0b1` to a bit sets the corresponding bit in the CH_MSK_ST register. Writing `0b0` has no effect. Each bit always reads as `0b0`.	WO	`0x0000_0000`

Channel Mask Clear (CH_MSK_CLR)

This register clears bits in the channel mask. This register is part of the Receiver channel window.

Table A-11 Channel Mask Clear register description

Bit	Name	Description	Type	Reset
[31:0]	FLAG_MSK_CLR{x}	Channel flag mask clear. Write `0b1` to a bit clears the corresponding bit in the CH_MSK_ST register. Writing `0b0` has no effect. Each bit always reads as `0b0`.	WO	`0x0000_0000`

Channel Interrupt Status (CH_INT_ST)

The following table describes the Channel Interrupt Status register.

Table A-12 Channel Interrupt Status register description

Bit	Name	Description	Type	Default
[31:1]	-	Reserved.	RO	`0x0000_0000`
[0]	CH_CLR	Channel status clear interrupt status. `0b0` – No Channel clear interrupt has occurred. `0b1` – A Channel clear interrupt has occurred.	RO	`0b0`

This field is:

- Set to `0b1` when the Receiver writes to CH_CLR registers

  Note

  If the Sender domain is not powered on, or is held in reset at the point the write occurs, then there is no requirement for the field to be set to `0b1`.

- Set to `0b0` when the Sender writes `0b1` to the CH_INT_CLR.CH_CLR field

The setting to `0b1` takes priority of setting it to `0b0`.

The value of the CH_INT_EN.CH_CLR has no effect on the value of CH_INT_ST.

Channel Interrupt Clear (CH_INT_CLR)

The following table describes the Channel Interrupt Clear register.

Table A-13 Channel Interrupt Clear register description

Bit	Name	Description	Type	Reset
[31:1]	-	Reserved	RO	`0x0000_0000`
[0]	CH_CLR	Clear Channel status clear interrupt. Write `0b1` to clear the interrupt. Writing `0b0` has no effect. This field always reads as `0b0`.	WO	`0b0`

Channel Interrupt Enable (CH_INT_EN)

The following table describes the Channel Interrupt Enable register.

Table A-14 Channel Interrupt Enable register description

Bit	Name	Description	Type	Reset
[31:1]	-	Reserved	RO	`0x0000_0000`
[0]	CH_CLR	Channel status clear interrupt enable. `0b0` – Channel clear interrupt disabled. `0b1` – Channel clear interrupt enabled.	RW	`0b0`

Message Handling Unit Configuration (MHU_CFG)

The following table describes the Message Handling Unit Configuration register.

Table A-15 Message Hanlding Unit Configuration register description

Bit	Type	Default	Name	Description
[31:7]	RO	0x000_0000	-	Reserved
[6:0]	RO	CFG_DEF	NUM_CH	Number of Channels: 0x00: Reserved 0x01: 1 Channel 0x02: 2 Channels … 0x7C: 124 Channels 0x7D: Reserved 0x7E: Reserved 0x7F: Reserved

Response Configuration (RESP_CFG)

The following table describes the Response Configuration register.

Table A-16 Response Configuration register description

Bit	Type	Default	Name	Description
[31:1]	RO	0x0000_0000	-	Reserved
[0]	RW	0b0	NR_RESP	Response generated when the Sender attempts to access any register of a Channel, when ACCESS_READY.ACC_RDY is 0b0. 0b0: Access is treated as RAZ/WI with no error generated. 0b1: Access is treated as RAZ/WI and an error is generated.

Access Request (ACCESS_REQUEST)

The following table describes the Access Request register.

Table A-17 Access Request register description

Bit	Type	Default	Name	Description
[31:1]	RO	0x0000_0000	-	Reserved
[0]	RW	0b0	ACC_REQ	Access Request: 0b0: Receiver is not requested to be ready to receive transfer 0b1: Receiver is requested to be ready to receive transfer

Access Ready (ACCESS_READY)

The following table describes the Access Ready register.

Table A-18 Access Ready register description

Bit	Type	Default	Name	Description
[31:1]	RO	0x0000_0000	-	Reserved
[0]	RO	0b0	ACC_RDY	Access Ready 0b0: Receiver is not ready to receive transfer 0b1: Receiver is ready to receive transfer

Interrupt Status (INT_ST)

The following table describes the Interrupt Status register.

Table A-19 Interrupt Status register description

Bit	Type	Default	Name	Description
[31:3]	RO	0x0000_0000	-	Reserved
[2]	RO	0b0	CHCOMB	Channel combined interrupt status: 0b0: No interrupt has occurred on any Channel 0b1: An interrupt has occurred on at least one Channel There is no corresponding bit in the INT_CLR register. To clear this interrupt, software must clear the underlying interrupt.
[1]	RO	0b0	R2NR	Ready to not ready interrupt status: 0b0: Ready to not ready interrupt has not occurred 0b1: Ready to not ready interrupt has occurred
[0]	RO	0b0	NR2R	Not ready to ready interrupt status: 0b0:Not ready to ready interrupt has not occurred 0b1: Not ready to ready interrupt has occurred

Interrupt Clear (INT_CLR)

The following table describes the Interrupt Clear register.

Table A-20 Interrupt Clear register description

Bit	Type	Default	Name	Description
[31:2]	RO	0x0000_0000	-	Reserved
[1]	WO	0b0	R2NR	Clear ready to not ready interrupt: Write 0b1 to clear the interrupt. Writing 0b0 has no effect.
[0]	WO	0b0	NR2R	Clear not ready to ready interrupt: Write 0b1 to clear the interrupt. Writing 0b0 has no effect.

Interrupt Enable (INT_EN)

The following table describes the Interrupt Enable register.

Table A-21 Interrupt Enable register description

Bit	Type	Default	Name	Description
[31:3]	RO	0x0000_0000	-	Reserved
[2]	RW	0b1	CHCOMB	Channel combined interrupt enable: 0b0: Combined interrupt is disabled 0b1: Combined interrupt is enabled
[1]	RW	0b0	R2NR	Ready to not ready interrupt enable: 0b0: Disables the generation of ready to not ready interrupt 0b1: Enables the generation of ready to not ready interrupt
[0]	RW	0b0	NR2R	Not ready to ready interrupt enabled 0b0: Disables the generation of not ready to ready interrupt 0b1: Enables the generation of not ready to ready interrupt

Channel Combined Interrupt Status {0-3} (CHCOMB_INT_ST{0-3})

The following table describes the Channel Combined Interrupt Status register.

The CHCOMB_INT_ST{0-3} registers are located at 0xFA0 to 0xFAC in the Sender frame, starting with CHCOMB_INT_ST0 at 0xFA0 and continuing in ascending order.

Table A-22 Channel Interrupt Stauts {0-3} register description

Bit	Type	Default	Name	Description
[31:0]	RO	0x0000_0000	CH_INT_ST{x}	Channel interrupt status. Each bit indicates whether a Channel has a pending interrupt or not. CHCOMB_INT_ST0 has the status for Channels 0 to 31, starting with Channel 0 at bit 0. CHCOMB_INT_ST1 has the status for Channels 32 to 63, starting with Channel 32 at bit 0. CHCOMB_INT_ST2 has the status for Channels 64 to 95, starting with Channel 64 at bit 0. CHCOMB_INT_ST3 has the status for Channels 96 to 123, starting with Channel 96 at bit 0. A bit relating to an unimplemented Channel is Reserved and treated as RAZ/WI. CHCOMB_INT_ST3 bits[31:28] are always Reserved and treated as RAZ/WI.

The fields within the CH_INT_ST{0-3} registers of the Sender frame, are set to 0b1 when the associated CH_INT_ST.CH_CLR field is 0b1.

Implementer Identification Register (IIDR)

The following table describes the Implementer Identification Register.

Table A-23 Implementer Identification Register description

Bit	Type	Default	Name	Description
[31:20]	RO	0x076	PRODUCT_ID	MHU part ID
[19:16]	RO	0x0	VARIANT	Major revision of the MHU

Table A-23 Implementer Identification Register description (continued)

Bit	Type	Default	Name	Description
[15:12]	RO	0x0	REVISION	Minor revisions of the MHU
[11:0]	RO	0x43B	IMPLEMENTER	Contains the JEP106 code for Arm: • [11:8] JEP106 continuation code of implementer. • [7] Always 0 • [6:0] JEP106 identity code of implementer.

Architecture Identification Register (AIDR)

The following table describes the Architecture Identification Register.

Table A-24 Architcture Identification Register description

Bit	Type	Reset	Name	Description
[31:8]	RO	0x00_0000	-	Reserved
[7:4]	RO	0x1	ARCH_MAJOR_REV	MHU Architecture Major Revision: 0x0: Reserved 0x1: MHUv2
[3:0]	RO	0x1	ARCH_MINOR_REV	MHU Architecture Minor Revision: 0x0: Minor Revision 0 0x1: Minor Revision 1 All other values are reserved.

Note

When ARCH_MAJOR_REV reads as 0x0, the values in the ARCH_MINOR_REV and IIDR fields are RAZ. Software must use a platform-specific way to determine the version of the MHU architecture that the component conforms to.

Peripheral ID 4 (PID4)

The following table describes the Peripheral ID 4 register.

Table A-25 Peripheral ID 4 register description

Bits	Type	Name	Reset	Description
[31:8]	RO	-	0x00_0000	Reserved
[7:4]	RO	Size	0x0	Number of 4KB occupied by the System ID block. This field is deprecated.
[3:0]	RO	DES_2	0x4	JEP Continuation

Peripheral ID 5 (PID5)

The following table describes the Peripheral ID 5 register.

Table A-26 Peripheral ID 5 register description

Bits	Type	Reset	Name	Description
[31:0]	RO	0x0000_0000	-	Reserved

Peripheral ID 6 (PID6)

The following table describes the Peripheral ID 6 register.

Table A-27 Peripheral ID 6 register description

Bits	Type	Reset	Name	Description
31:0	RO	0x0000_0000	-	Reserved

Peripheral ID 7 (PID7)

The following table describes the Peripheral ID 7 register.

Table A-28 Peripheral ID 7 register description

Bits	Type	Reset	Name	Description
[31:0]	RO	0x0000_0000	-	Reserved

Peripheral ID 0 (PID0)

The following table describes the Peripheral ID 0 register.

Table A-29 Peripheral ID 0 register description

Bits	Type	Reset	Name	Description
[31:8]	RO	0x00_0000	-	Reserved
[7:0]	RO	0x76	PART_0	Bits [7:0] of part ID

Peripheral ID 1 (PID1)

The following table describes the Peripheral ID 1 register.

Table A-30 Peripheral ID 1 register description

Bits	Type	Reset	Name	Description
[31:8]	RO	0x00_0000	-	Reserved
[7:4]	RO	0xB	DES_0	Bits [3:0] of JEP 106 Identity
[3:0]	RO	0x0	PART_1	Bits [11:8] of part ID

Peripheral ID 2 (PID2)

The following table describes the Peripheral ID 2 register.

Table A-31 Peripheral ID 2 register description

Bits	Type	Reset	Name	Description
[31:8]	RO	0x00_0000	-	Reserved
[7:4]	RO	0x0	REVISION	Major revision of the System ID block

Table A-31 Peripheral ID 2 register description (continued)

Bits	Type	Reset	Name	Description
[3]	RO	0b1	JEDEC	Indicates the use of JEDEC JEP106 identification scheme
[2:0]	RO	0b011	DES_1	Bits [6:4] of JEP 106 Identity

Peripheral ID 3 (PID3)

The following table describes the Peripheral ID 3 register.

Table A-32 Peripheral ID 3 register description

Bits	Type	Reset	Name	Description
[31:8]	RO	0x00_0000	-	Reserved
[7:4]	RO	0x0	REVAND	Minor revision of the System ID block
[3:0]	RO	0x0	CMOD	Customer modification field

Component ID 0 (CID0)

The following table describes the Component ID 0 register.

Table A-33 Component ID 0 register description

Bits	Type	Reset	Name	Description
[31:8]	RO	0x00_0000	-	Reserved
[7:0]	RO	0x0D	PRMBL_0	Preamble 0

Component ID 1 (CID1)

The following table describes the Component ID 1 register.

Table A-34 Component ID 1 register description

Bits	Type	Reset	Name	Description
[31:8]	RO	0x00_0000	-	Reserved
[7:4]	RO	0xF	CLASS	Class of the component
[3:0]	RO	0x0	PRMBL_1	Preamble 1

Component ID 2 (CID2)

The following table describes the Component ID 2 register.

Table A-35 Component ID 2 register description

Bits	Type	Reset	Name	Description
[31:8]	RO	0x00_0000	-	Reserved
[7:0]	RO	0x05	PRMBL_2	Preamble 2

Component ID 3 (CID3)

The following table describes the Component ID 3 register.

Table A-36 Component ID 3 register description

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved.	RO	0x00_0000
[7:0]	PRMBL_3	Preamble 3.	RO	0xB1

A.2.6 Limitations

The Message Handling Unit only guarantees that messages are received by the Receiver if:

- Ready to Send protocol is used.
- Both the Sender and Receiver complete the required steps of the Transport Protocols, as defined in *A.3 Transport protocols* on page Appx-A-270.
- Both the Sender and Receiver are not reset during the process.

A.3 Transport protocols

There are several ways in which the MHUv2 can be used to facilitate the communication between two entities.

This chapter does not define the messages protocols, that is, the format of the messages. However, the MHUv2 (*A.2 Message Handling Unit v2* on page Appx-A-254) does define several transport protocols, the methods that are used to transfer the messages. There may be other transport protocols, which can be implemented using MHUv2, but only the ones that are documented in this chapter are guaranteed to be supported by any future version of the MHUv2.

——— **Note** ———

In Appendix A, the transport protocols can transfer any message protocol, even messages which are made up of of several transfers.

The guarantee only extends to MHUv2.x. There is no guarantee that the same methods will be supported in a future MHUvX specification, where X is greater than 2.

———————————

This section contains the following subsections:

- *A.3.1 Doorbell* on page Appx-A-270.
- *A.3.2 Single-word transfer* on page Appx-A-271.
- *A.3.3 Multi-word transfer* on page Appx-A-271.

A.3.1 Doorbell

The doorbell transport protocol uses MHUv2 to generate an interrupt to the Receiver only.

Each flag bit of the CH_ST register is used to indicate when a transfer is being sent by the Sender. When the Sender has sent a transfer, it can send other transfers using other flags within the same Channel, or any flag within another Channel, but must not use the same flag bit until the Receiver has processed the transfer.

It is dependent on the message protocol whether there is data associated with the flag, transferred in out-band memory, or whether the flag bit indicates the full message. If there is out-band data, then the location of the data is pre-agreed between the two entities. The following figure shows the sequence of events to implement the doorbell transport protocol.

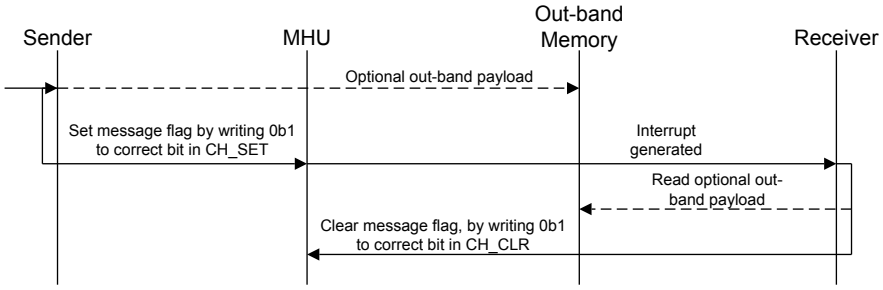

Figure A-1 Sequence of events for doorbell transport protocol

1. Sender writes the optional payload for the transfer to the out-band memory. The location is agreed using a software IMPLEMENTATION DEFINED method.
2. Sender writes `0b1`, to the correct bit in the CH_SET register.
3. MHU generates the interrupt to the Receiver.
4. Receiver gets the interrupt and reads the transfer payload in the out-band memory, if there is a transfer payload.

5. Receiver clears the flag by writing `0b1`, to the corresponding bit in the CH_CLR register.
6. Sender must wait for the bit in the CH_ST register, set to `0b1`, to become `0b0` to know when the transfer has been processed by the Receiver. This can be done by either:
   - Polling the CH_ST register, checking for the bit set to `0b1` to become `0b0`
   - Waiting to receive a clear interrupt for that Channel, and check that the bit has transitioned to `0b0`

This protocol has the following requirements:

- If there is an out-band memory used, then the location has been agreed between Sender and Receiver and is accessible by both.
- Any CH_ST flag bits which are used to indicate a transfer, must not be masked in the Receiver. I.e. if bits [3:0] are used to indicate 4 different transfers, bits [3:0] of CH_MSK_ST must be `0b0`.

A.3.2 Single-word transfer

Each transfer is a single word transferred in-band, using a single Channel.

There can be multiple transfers to send a single message and each transfer can be any value, except for 0. When the Sender has sent the transfer, the Receiver processes the transfer. Until the Receiver has processed the transfer, the Sender is not allowed to use that Channel to send another transfers. While the Sender, is waiting for the Receiver to process the transfer, it can use other Channels to send other transfers. The following figure shows the required sequence to implement the single-word transfer transport protocol.

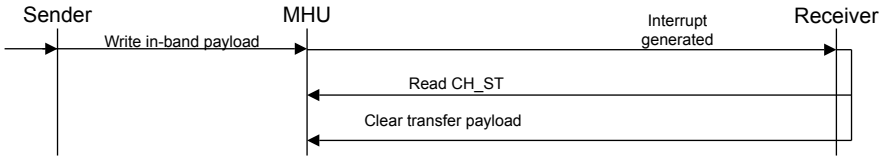

Figure A-2 Sequence of events for single-word transfer transport protocol

1. Using the CH_SET register, the sender writes the in-band payload for the transfer,. The in-band payload of the transfer can be any value, except all 0s.
2. MHU generates the interrupt to the Receiver.
3. Receiver receivers the interrupt and reads the in-band payload for the transfer, using the CH_ST register.
4. Receiver clears the in-band payload for the transfer by writing the `0b1` to each bit of the CH_CLR register.
5. To know when the transfer has been processed by the Receiver, the sender must wait for all bits in the CH_ST register to read as `0b0`. This can be done by either:
   a. Polling the CH_ST register, checking for the register to read as all 0s.
   b. Waiting to receive a clear interrupt for that Channel, and checking that all bits read as `0b0`.

This protocol has the following requirements:

- All transfer payloads must have at least one bit set to `0b1`.
- The CH_MSK_ST register must have at least one bit set to `0b0`.
- At least one bit, which is set to `0b1`, in the CH_ST and the corresponding bit in the CH_MSK_ST set to `0b0`.

Arm strongly recommends, that software either:

- Has all CH_MSK_ST bits set to `0b0`
- A single bit is used to indicate a transfer to the Receiver, and only that bit is required to be `0b0` in the CH_MSK_ST. For example, if bit [31] is used to indicate a transfer, only CH_MSK_ST[31] needs to be set to `0b0`.

A.3.3 Multi-word transfer

Each transfer is made of two or more words.

The Sender writes each word of the transfer, but only the final word of the transfer is used to indicate to the Receiver, that a transfer is waiting. For this transport protocol, the number of Channels that are used

can be 1 or more. It is not a requirement that all the words are transferred using Channels of the MHU, some words of the transfer can be in out-band memory. The location of the out-band memory can be either pre-agreed or can be referenced as part of the transfer payload. The following figure shows the sequence of events used to implement the multi-word transfer transport protocol.

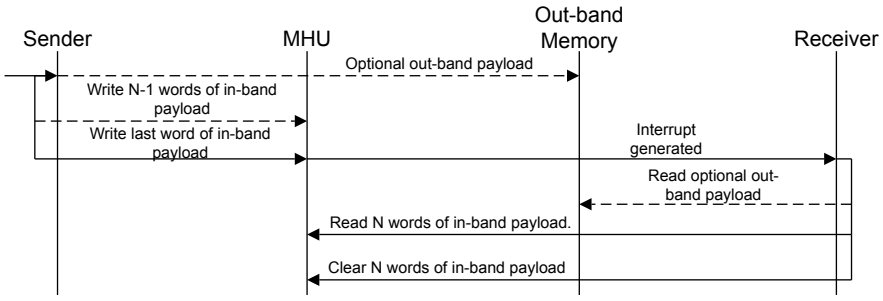

Figure A-3 Sequence of events for multi-word transfer transport protocol

1. Sender writes the optional out-band payload to a location in out-band memory. The location in the out-band memory is either:
   - Pre-agreed using a software IMPLEMENTATION DEFINED method
   - Pointed to via the payload transferred using the Channel of the MHU
2. Sender writes N-1 words of the in-band transfer payload to the Channels, using the CH_SET register for each Channel. The values of these Channels is allowed to be zero.
3. Sender writes the last word of the in-band transfer to the final Channel. This value must be nonzero.
4. MHU generates the interrupt to the Receiver.
5. Following reception of the interrupt, the receiver then:
   a. Reads optional out-band payload, if provided. The method which the Receiver knows where there is an out-band is software IMPLEMENTATION DEFINED.
   b. Reads in-band payload, using the CH_ST register for each Channel.
6. Receiver clears the in-band payload from all Channels by writing `0b1` to each bit of the CH_CLR register for each Channel.
7. Sender waits for all the CH_ST registers of the used Channels to read as all 0s, to know that the transfer has been processed by the Receiver. This can be done by either:
   a. Polling all CH_ST registers of Channels used by the transfer- to read as all 0s
   b. Waiting for a clear interrupt for each Channel used, and then checking that the CH_ST register reads as all 0s

It is software IMPLEMENTATION DEFINED:

- The order in which the Sender writes the out-band and N-1 words of in-band payload is software dependant. However, the last word of in-band payload must only be written, after all in-band and out-band payload data is guaranteed to be visible to the Receiver.
- The order in which the Receiver reads the out-band and in-band payload
- The order in which the Receiver clears the in-band payload

This protocol has the following requirements:

- CH_MSK_ST for Channels, which are not the last written word of the transfer, have all bits set to `0b1`.
- CH_MKS_ST for Channel, used to transfer the last written word, must have at least one bit set to `0b0`.
- At least one bit, which is set to `0b1`, in the CH_ST and the corresponding bit in the CH_MSK_ST set to `0b0`.
- The order in which words of the transfer are written, by the Sender, can be in any order so long as the Channel which has at least one bit of the CH_MSK_ST register set to `0b0`, is the last word to be written. The last word to be written to a Channel, does not need to be the last word of the transfer, it

can be the first word. That is, the Sender writes words 1, 2 and 3 of a 4-word transfer first, then writes word 0.
- If out-band memory is required, the location of the memory must accessible by both the Sender and Receiver.

Arm recommends:
- The last Channel cleared by the Receiver, is the Channel with at least one bit of the CH_MSKST register set to `0b0`. This allows the Sender to wait for only that CH_ST register to read as all 0s, by either polling or waiting for the clear interrupt for that Channel.

Note

If the number of words in a transfer is greater than the number of Channels allocated, the transfer process is the same as *A.3.2 Single-word transfer* on page Appx-A-271 shows, except:
- Sender must write all words of transfer payload, which is stored in out-band memory, before writing the last word of transfer payload to the MHU Channel.
- Receiver must read all of the transfer payload, included words stored in out-band memory, before clearing the Channels in the MHU. There is no requirement to clear the words in the out-band memory

Appendix B
Interrupt Router

This appendix describes the Interrupt Router.

The Interrupt Router is a programmable router for interrupts, located before the *Interrupt Controllers* (ICs) in SSE-700. Software can select which ICs a specific shared interrupt is forwarded to.

IC can refer to either of the following:

- GIC-400 in Host System
- Interrupt controller in External System 0 and 1
- Interrupt collator in Secure Enclave

Interrupt Router in SSE-700 only supports routing of *Shared Peripheral Interrupt* (SPI). It is the signal based interrupt that can be routed to any of the interrupt controllers in an SSE-700 based SoC.

It contains the following sections:

B.1 Overview

The following figure shows the SSE-700 Interrupt Router block diagram.

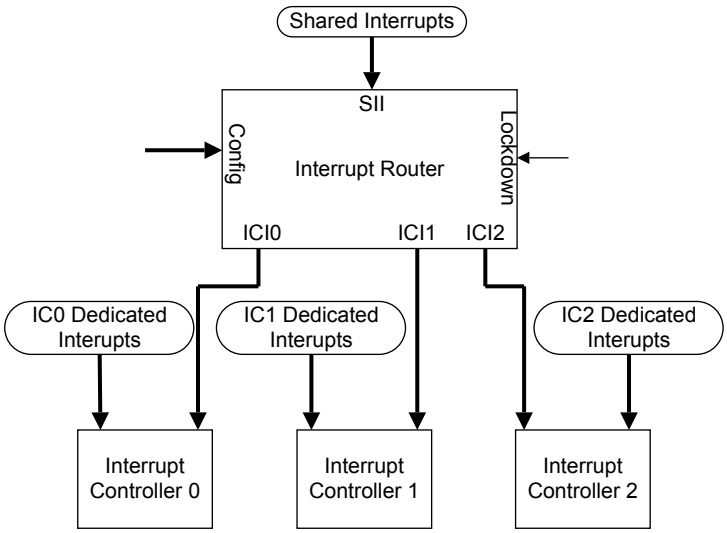

Figure B-1 SSE-700 Interrupt Router

The Interrupt Router has the following configuration options:

- Number of shared interrupts (NUM_SHD_INT) supported by the Interrupt Router. This must be in the range of 0 to 395.
- Interrupt Controller Interrupt Mask (SI{x}_ICI_DST) per each shared interrupt, where x is the shared interrupt number between 0 and NUM_SHD_INT-1.
- Interrupt Default Interrupt Controller Interface (SI{x}_DEF_ICI) per each shared interrupt, where x is the shared interrupt number between 0 and NUM_SHD_INT-1.

Note

The Interrupt Router can be configured so that a shared interrupt can only be routed to a single ICI interface.

When this specification refers to a shared interrupt number, this is not to be confused with the final interrupt ID of the interrupt of the target Interrupt Controller. Which can be different for each Interrupt Controller the interrupt can be routed to. For example, an interrupt, with shared interrupt number 0, can have an interrupt ID of 32, 123 and 4 at different Interrupt Controllers within a SSE-700 based SoC.

B.2 Software configuration of Interrupt Router

The Interrupt Router provides a window-based configuration scheme using the SHD_INT_{INFO,CFG,LCTRL} registers.

The SHD_INT_SEL register allows software to select which interrupt the SHD_INT_{INFO,CFG,LCTRL} registers are referring to. If the SHD_INT_SEL register selects an interrupt which is not implemented, the SHD_INT_{INFO,CFG,LCTRL} registers are Reserved and treated as RAZ/WI.

The SHD_INT_INFO.ICI_DST field indicates which ICI interfaces the interrupt can be routed to. The SHD_INT_INFO.ICI_DST field is set to the value of the SI{x}_ICI_DST configuration option for the shared interrupt selected by the SHD_INT_SEL.INT_SEL field.

The SHD_INT_CFG.ICI_EN field configures which ICI interfaces the interrupt is routed to. The default value of this field depends on the SI{x}_DEF_ICI configuration option for the shared interrupt selected by the SHD_INT_SEL.INT_SEL field. It is UNPREDICTABLE whether the interrupt is routed to the correct destinations if software changes the value in the SHD_INT_CFG.ICI_EN field when the interrupt source is active.

——— **Note** ———

Arm strongly recommends that software disables the interrupt at source, before making any changes to the SHD_INT_CFG.ICI_EN field.

———————————

The SHD_INT_LCTRL.LOCK field configures whether the interrupt's configuration is locked or not. For more information, see *B.4 Lockdown extension* on page Appx-B-278.

B.3 Interrupt routing

Interrupts received by the Interrupt Router are routed to the connected Interrupt Controllers based on:

- Subset of allowed Interrupt Controllers, a shared interrupt can be routed to. This is defined by SI{x}_ICI_DST.
- Software configuration for the shared interrupt

The following figure shows how a shared interrupt is routed. Which ICI the interrupt is routed to, from the allowed ICIs, is selected using the SHD_INT_CFG.ICI_EN field.

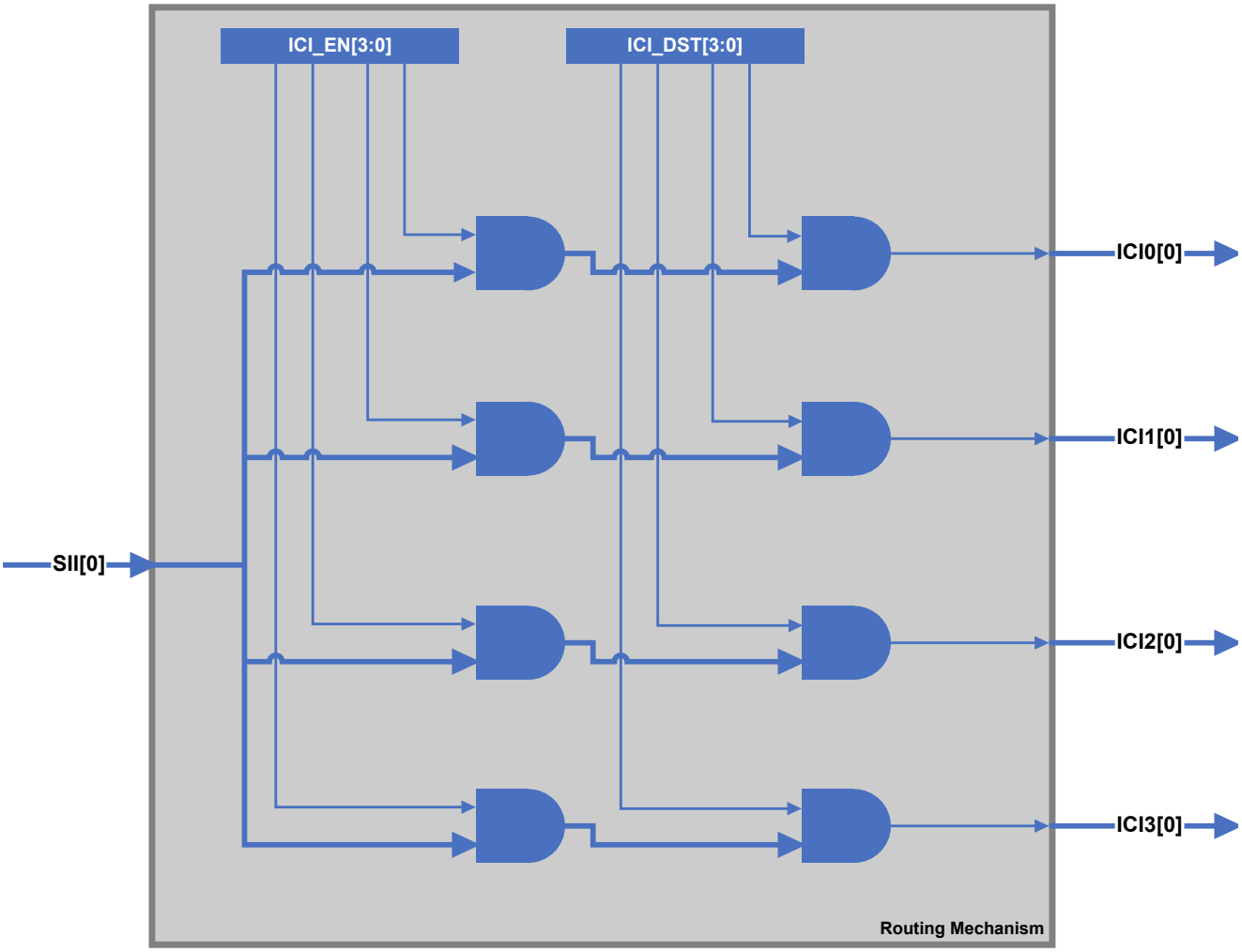

Figure B-2 Shared interrupt routing

Note

It is possible for a shared interrupt to be routed to more than one Interrupt Controller at the same time.

B.4 Lockdown extension

The Interrupt Router supports the ability to lockdown the configuration of the Interrupt Router.

Software can lock down an individual shared interrupt configuration or all shared interrupt configurations, depending on the level of the Lockdown Extension which the implementation of the Interrupt Router supports.

SSE-700 supports Lockdown Extension Level 2. Lockdown is supported at the individual shared interrupt and Interrupt Router level.

——— **Note** ———

In this manual, the level of the Lockdown Extension implemented by the Interrupt Router is sometimes referred to using the shorthand: LDE.2.

———————————

This section contains the following subsections:

B.4.1 Interrupt Router lockdown states

The following sections describe the Interrupt Router lockdown states.

Open

All registers can be updated, except for:

- SHD_INT_CFG when the SHD_INT_LCTRL.LOCK field is `0b1`

Partial

All registers can be updated, except for:

- SHD_INT_CFG when the SHD_INT_LCTRL.LOCK field is `0b1`
- SHD_INT_LCTRL.LOCK field, which is set to `0b1`
- INT_RTR_CTRL

Full

The following registers are read-only:

- SHD_INT_CFG
- SHD_INT_LCTRL

The LD_CTRL.LOCK field is read-only if the Lockdown interface is asserted.

The Interrupt Router implements a Lockdown interface which is a single signal which, when asserted, prevents the Interrupt Router lockdown state from being changed, when it is Partial or Full. It is driven by the INT_RTR_LOCK bit of Host System Lock Control registers. See section *Host System Lock Control Status (HOST_SYS_LCTRL_ST) register* on page 11-187 for details.

B.4.2 Tamper Interrupt

The Interrupt Router reports Tamper Interrupts using a Tamper Interrupt interface.

The interface is made up of a single level sensitive interrupt to Secure Enclave. It is asserted when either INT_RTR_TMP_ST.TMP_ST_VLD or INT_RTR_TMP_ST.TMP_SWT_OVERFLOW bits are `0b1`.

For details about the Tamper Interrupt mapping of the Interrupt Router, see *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual*.

B.4.3 Configuration Access and Lockdown

An access to the configuration registers of the Interrupt Router are called Configuration Accesses.

It becomes a Configuration Access Error if any of the following occurs:

- Update the configuration of an interrupt which is locked (SHD_INT_LCTRL.LOCK set to `0b1`).
- Unlock a locked interrupt (set SHD_INT_LCTRL.LOCK to `0b0` when it is `0b1`) when the Interrupt Routers Lockdown state is Partial or Full.
- Change the Interrupt Routers Lockdown state, when it is Partial or Full, and the Lockdown interface is asserted, except for the value of SHD_INT_SEL register.
- Update the value of INT_RTR_CTRL register.
- Accesses the INT_RTR_TMP_ST register with a transaction that does not have the properties as described in *B.4.4 Tamper report access* on page Appx-B-279.

When any of the above occur the Interrupt Router does the following:

- Generates a Configuration Access Error.
- Asserts the Tamper Interrupt (if it is not already asserted) and does either of the following:
  - — Generates a tamper report, updating the INT_RTR_TMP_ST register with the address of the Configuration Access It also sets the INT_RTR_TMP_ST.TMP_ST_VLD field to `0b1`, if the INT_RTR_TMP_ST.TMP_ST_VLD was `0b0`.
  - — Generates a tamper report overflow, setting the INT_RTR_TMP_ST.TMP_ST_OVEFLW bit to `0b1`, if INT_RTR_TMP_ST.TMP_STP_VLD is `0b1`.

For more information on Configuration Accesses and Configuration Access Errors, see *B.5 Configuration Accesses* on page Appx-B-280.

B.4.4 Tamper report access

The INT_RTR_TMP_ST holds information about any transactions that attempt to modify configurations of an Interrupt Router in a state of Lockdown.

Access to the INT_RTR_TMP_ST register is only allowed by transactions with the following properties:

- Secure privileged transactions
- From a Secure Monitor agent

If an access is attempted without either of the mentioned properties, the Interrupt Router does the following:

- Generates a Configuration Access Error
- Asserts the Tamper Interrupt, if it is not already asserted, and does either of the following:
  - — Generates a tamper report, updating the INT_RTR_TMP_ST register with the address of the Configuration Access and set the INT_RTR_TMP_ST.TMP_ST_VLD field to `0b1`, if the INT_RTR_TMP_ST.TMP_ST_VLD was `0b0`
  - — Generates a tamper report overflow, setting the INT_RTR_TMP_ST.TMP_ST_OVEFLW bit to `0b1`, if INT_RTR_TMP_ST.TMP_STP_VLD is `0b1`

Secure Monitor

A Secure Monitor agent is the most trusted master in the SoC and the Tamper Interrupt is routed to the interrupt controller associated by the Secure Monitor.

B.5 Configuration Accesses

Access to the configuration registers of the Interrupt Router is called a Configuration Access.

A Configuration Access can become a Configuration Access Error if it does any of the following:

- Access a Reserved location
- Access is not a 32-bit word aligned access
- Performs any of the actions described in *B.4.3 Configuration Access and Lockdown* on page Appx-B-279

When a Configuration Access Error occurs, the Interrupt Router must do the following:

- Generate an error response to the Configuration Access, if the INT_RTR_CTRL.ERR field is set to `0b1`. Otherwise, the response to the Configuration Access completes without an error.
- Read data is set to all zeros
- Write data is ignored

Note

A Configuration Access Error does not cause a Tamper Interrupt to be generated. A Tamper Interrupt is only generated if one of the conditions defined in *B.4.3 Configuration Access and Lockdown* on page Appx-B-279 occur.

Appendix C
Firewall

This appendix describes the Firewall.

It contains the following sections:

C.1 About the Firewall

This section gives a brief introduction to the Firewall.

Between different entities within the SoC, both monitoring and protection of the address space is required.

——— **Note** ———

In this chapter, an entity is either:

- A bus master
- A software application executing on a bus master

———————————

The ability to compartmentalize the address space of a system between different entities provides protection between the entities. This compartmentalization also protects an entity from performing operations that are not allowed within its own compartment.

There is also the need to perform translation from one address space to another. For example, an external system needs to access the Host System memory map. If the external system has its own dissimilar memory map, it may not be possible for the external system to access locations in the Host System due to memory map address clashes between the systems. Address translation provides an access window from the external system to the Host System, where an appropriate shared resource resides. The address from the external system is treated like a virtual address and is translated into a physical address in the Host System.

Compartmentalization is achieved through various methods, including traditional Memory Management Units (MMU). However, the memory footprint overhead in supporting page table descriptors is prohibitive in a memory constrained device. The SSE-700 Firewall is designed to provide some of the benefits of traditional memory management without the associated overhead of page table descriptors.

The monitoring functionality detects when accesses are being made that cause errors to be generated in the system. The detection of an error can be indicative of an entity making illegal accesses. Detection and reporting of these errors, provides the ability for software to take corrective action or to limit the impact, if the errors are generated by a prohibited action.

This section contains the following subsections:

- *C.1.1 Firewall usage* on page Appx-C-282.
- *C.1.2 Extensions* on page Appx-C-285.
- *C.1.3 Bus protocol* on page Appx-C-287.

C.1.1 Firewall usage

This section shows an example usage of the Firewall.

The Firewall can be used as an address space protection and monitoring unit for memory and for peripherals in a system, as the following figure shows. The Firewall block represents all Firewall Components and is placed between the master, memory and peripherals.

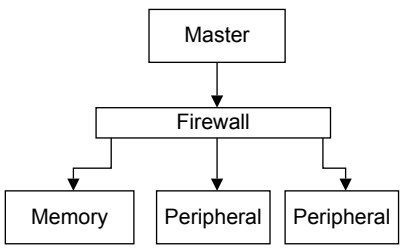

Figure C-1 Example of using Firewall as protection and monitoring for a single master

The Firewall can also be used to monitor and protect the memory and peripherals from multiple masters as the following figure shows.

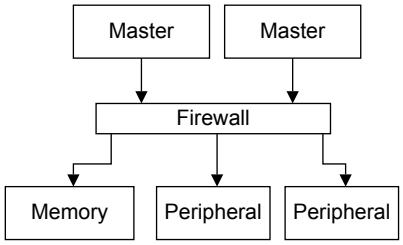

Figure C-2 Example of using Firewall as protection and monitoring for multiple masters

The Firewall can also be used with Firewall Components either side of an interconnect. This allows Firewall Components after the interconnect (FC2-4 in the figure below) to define regions that are defined by Firewall Components before the interconnect (FC1 in the figure below). By allowing refinement of regions, the granularity of how the address space is divided between masters in the system can be retained without increasing the number of regions that are required in each Firewall Component.

The following figure shows an example system using Firewall Components both sides of the interconnect: a CPU and two masters, A and B, attached to two peripherals, 0 and 1, and a small amount of memory. They are connected via an interconnect. This provides an error response to any transaction that targets a location that is not allocated to a downstream device.

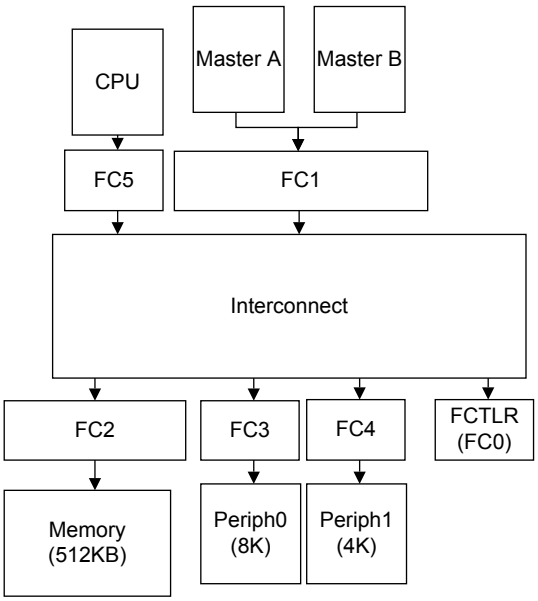

Figure C-3 Example system with Firewall Components both sides of the interconnect

In this example, the following table shows the levels of extensions that the Firewall Components implement.

Table C-1 Example Firewall Component levels of extensions

Firewall Component	PE level	ME level	TE level
FC0	1	0	0
FC1	2	2	2
FC2	2	0	0
FC3	1	0	0
FC4	1	0	0
FC5	0	2	0

The Firewall Components can implement any level of extension, but for this example this has no effect on the behavior.

Masters A and B can be either masters, for example DMA or Display, or can be systems with their own address space. For this example, both masters are systems with their own independent address spaces. Each of these exposes one or more windows into the address space of the Host System, in this case the CPU address.

The following figure shows example memory maps for each address space.

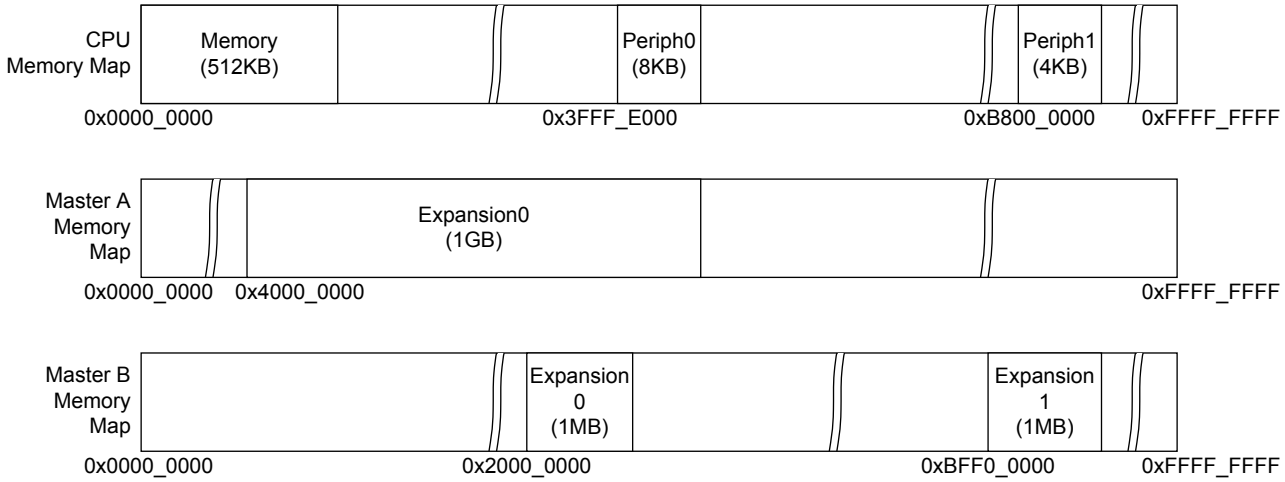

Figure C-4 Memory maps for CPU, Master A, and Master B

Transactions issued by Master A and B are not altered as they enter the Host System.

There are five Firewall Components in the system between the masters and peripherals, excluding the Firewall Controller.

In this example software divides the resources of the Host System, as the following table shows.

Table C-2 Example memory regions requirements of example system

Region identifier	Description	Base address	Size of region	Access masters and privileges
0	CPU Secure code and data	0x0000_0000	8KB	CPU - S RWX
1	CPU Non-secure code and data	0x0000_4000	16KB	CPU - NS RWX
2	Master A to CPU mailbox	0x0002_0000	4KB	CPU - NS R Master A - NS RW
3	CPU to Master A mailbox	0x0002_1000	4KB	CPU - NS RW Master A - NS R
4	Master A Non-secure data	0x0002_2000	8KB	Master A - NS RWX
5	Master B to CPU mailbox	0x0003_0000	4KB	CPU - NS R Master B - NS RW
6	CPU to Master B mailbox	0x0003_1000	4KB	CPU - NS RW Master B - NS R
7	Master B Non-secure data	0x0003_2000	8KB	Master B - NS RWX
8	Peripheral 0 allocated to Master A	0x7FFF_E000	8KB	Master A - NS RW
9	Peripheral 1 allocated to Master B	0xB800_0000	4KB	Master B - NS RW

In the table above, the Region Identifier column is not to be confused with the Region ID or Region Number, which is used in other sections of this chapter.

To achieve this allocation of resources, the following regions must be defined in the Firewall Components, as the following table shows.

Table C-3 Example Region definition for example system

Firewall Component	Region number	Region base address	Region size	Master ID	Security	Access types	Translation enable	Output base address
FC1	1	0x4000_0000	1GB	Master A	NS	RWX	Yes	0x0000_0000
	2	0x2000_0000	1MB	Master B	NS	RWX	Yes	0x0003_0000
	3	0xBFF0_0000	1MB	Master B	NS	RW	Yes	0xB800_0000
FC2	1	0x0000_0000	8KB	CPU	S	RWX	N/A	-
	2	0x0000_4000	16KB	CPU	NS	RWX	N/A	-
	3	0x0002_0000	4KB	CPU	NS	R	N/A	-
				Master A	NS	RW	N/A	-
	4	0x0002_1000	4KB	CPU	NS	RW	N/A	-
				Master A	NS	R	N/A	-
	5	0x0002_2000	8KB	Master A	NS	RW	N/A	-
	6	0x0003_0000	4KB	CPU	NS	R	N/A	-
				Master B	NS	RW	N/A	-
	7	0x0003_1000	4KB	CPU	NS	RW	N/A	-
				Master B	NS	R	N/A	-
	8	0x0003_2000	8KB	Master B	NS	RW	N/A	-
FC3	0	N/A	N/A	Master A	NS	RW	N/A	-
FC4	0	N/A	N/A	Master B	NS	RW	N/A	-

Note

In the previous table above, the "Access masters and privileges" column uses the following format for describing the allowed access types from each master: {Master_Name} – {Security Type} – {Privilege}. The Security type can be either Secure (S) or Non-secure (NS). The privilege is a combination of one or more Reads (R), Writes (W), or Executes (X). If a master is not present in the column, then it has no access.

Firewall Components 0 and 5, are ignored in this example. FC0 only defines regions to protect the Firewall. For more information on Firewall Component 0 regions, see *Regions and RWE* on page Appx-C-373. FC5 is provided to monitor the bus transactions, as the CPU implements its own MMU.

Region 0 is not used for Firewall Components 1 and 2 as this is the default region, with different properties to other regions in the Firewall. For Firewall Components 3 and 4, region 0 is an ordinary region. For more information on the default region, see *Default regions* on page Appx-C-311.

C.1.2 Extensions

The Firewall architecture has several extensions which provide additional features for the Firewall.

An extension applies either at the individual Firewall Component, or the Firewall level. Each extension defines at least two levels of support, starting at level 0. Level 0 indicates the extension is not

implemented. An implementation of a Firewall can select the level to which each extension is implemented.

The following table gives a summary of each extension.

Table C-4 Extension summary table

Extension	Level	Description
Protection Extension (PE) on page Appx-C-305	0	No ability to compartmentalize the address space
	1	Ability to compartmentalize the address space, using predefined regions
	2	Ability to compartmentalize the address space, using software defined regions
Monitor Extension (ME) on page Appx-C-343	0	No ability to detect errors
	1	Ability to detect errors, but with limited information about the transaction
	2	Ability to detect errors, with full information about the transactions
Region Size Extension (RSE) on page Appx-C-364	0	Region's size must be of a power of 2, starting from the *Minimum Region Size* (MNRS)
	1	Region's size can be either an integer multiple of MNRS or a power of 2, starting from the MNRS
Translation Extension (TE) on page Appx-C-356	0	Output transaction has the same address and transaction properties as the incoming transaction
	1	Output transaction has the same address as the incoming transaction but can have different transaction properties
	2	Output transaction can have different address and transaction properties to the incoming transaction
Lockdown Extension (LDE) on page Appx-C-366	0	No ability to prevent update to configuration registers of the Firewall
	1	Ability to prevent updates to configuration registers at the Firewall Component level
	2	Ability to prevent updates to configuration registers at the region and Firewall Component levels
Security Extension (SE) on page Appx-C-365	0	Firewall only support a single security world
	1	Firewall Components support two security worlds (Secure and Non-secure) for regions, but only a single security world for configuration
	-	-
Save and Restore Extension (SRE) on page Appx-C-370	0	Firewall Component, when in the Disconnected state, lose all configuration and are inaccessible by software
	1	Firewall Component, when in the Disconnected state, are accessible by software. When a Firewall Component returns from the Disconnected state, its registers are restored automatically.

For more information on each extension see chapters *C.4 Protection Extension* on page Appx-C-305 to *C.10 Save and Restore Extension* on page Appx-C-370.

C.1.3 Bus protocol

This manual allows any bus protocol to be used on the interfaces to the Firewall Components. However, the bus protocol must meet the following requirements:

- Bus protocol must be either a burst or beat based protocol:
  - For a burst-based protocol, the range of bytes accessed by the transaction must be provided at the start of the transaction. An example of a burst-based protocol is AXI.
  - For a beat-based protocol, the range of bytes accessed by the current beat must be provided. An example, of a beat-based protocol is AHB.
- Provides an identifier of the master or group of masters, which issued the transaction, when more than one master or groups of masters are connected to the same Firewall Component. For more information see *Stream ID* on page Appx-C-288.
- Identifies whether the transaction is a read or a write.
- Identifies whether the transaction is Secure or Non-secure, if SE.1 or greater is implemented.
- Able to indicate an error response to the master who issued the transaction.
- Except for the address of the transaction the other properties of the transaction are the same throughout the transaction:
  - For a beat-based protocol the properties can change every beat.

This manual allows a single Firewall to implement different bus protocols or configurations of the same bus protocol, per each Bus Slave and Master interface. It is IMPLEMENTATION DEFINED how the Firewall Components handle the conversion between the different bus protocols and configurations.

Note

Arm recommends that if the bus protocol can indicate a decode error separate to a slave error, that the Firewall uses:

- A decode error for transactions which are blocked by the Firewall (this includes access to the Firewall's owner configuration registers).
- A slave error for any transaction which attempts to access the configuration registers of the Firewall and performs an operation which is not allowed.
- For transactions which the Firewall allows through the response should be returned unaltered.

Transaction properties

This section describes the transactions properties associated with transactions processed by the Firewall.

The Firewall has the following properties that it associates with transactions it receives or issues on its interfaces:

- Address: The address range of the transaction.
- Type: Whether the access is read or write.
- Instruction: Whether the access is instruction or data access. Only applies to read transactions, all write transactions are considered data accesses.
- Privilege: The privilege level of the transaction.
- Memory Attribute: The memory type, cache allocation policy and whether it is transient or not.
- Security: The security of the transaction.

A Firewall Component never issues an outgoing transaction which has illegal properties, whether this is because of illegal incoming properties or bad software programming when translation is being applied. When outputting the transaction, the following rules are obeyed when the Firewall Component implements PE.1 or greater:

- If a transaction has a memory attribute of Normal with either the Inner or Outer cache level being Non-cacheable, the transaction is considered to be read-no-allocate, write-no-allocate and non-transient at that level, regardless of the programmed or incoming values. A non-cacheable access is

considered read-no-allocate, write-no-allocate and non-transient. This is independent of programmed or incoming value.
- If a transaction has a memory attribute of Normal with a cacheable type that is read-no-allocate and write-no-allocate the memory has no transient attribute and is always considered non-transient independent of the programmed or incoming value.

Note

These requirements apply independently of:
- The level of TE implemented
- The value of the property's translation enable

For more information on memory attributes see *Output transaction memory attribute property on page Appx-C-357*.

When PE.0 is implemented transactions are always considered to be issued without going through the Firewall Component.

Stream ID

The StreamID identifies the master or group of masters in the SoC.

The StreamID is sent alongside all transactions using the **AxMMUSID** signal.

The StreamID is passed through Firewall Component, unaltered, alongside the transaction.

Single master or group of masters

This section describes how Firewall supports a single master or a single group of masters.

When a Firewall Component is connected to a single master or a single group of masters, the transactions that are issued to the Firewall Component are not required to indicate the MasterID. A Firewall Component supports being configured for use with a single master, FC_CFG2.SINGLE_MST set to `0b1`.

When a Firewall Component is configured to support a single master, it does the following:
- Ignores the incoming MasterID of transactions
- The following fields are hardwired to the same value, set at design time:
  - RGN_MID
  - FE_MID
  - EDR_MID
  - FW_TMP_MID, for the Firewall Controller only
- When outputting the transaction, the MasterID is set to the same value in the fields listed above.

C.2 Firewall interfaces

This section describes Firewall interfaces.

The Firewall has the following interfaces:

- One or more Bus Slave interfaces: transactions to be processed, by the Firewall Component, arrive on this interface.
- One or more Bus Master interfaces: transactions, which pass the checks performed by the Firewall Component, are output to the destination on this interface.
- One Programming interface: Firewall configuration accesses are received and then processed on this interface.
- One or more Power Control interfaces: used to indicate the power mode of the domain the Firewall Component(s) resides in.
- One or more Clock Control interfaces: used to indicate the status of the clock domain the Firewall Component(s) resides in.
- One Lockdown interface: used to provide configuration protection. Only present if LDE.1 or greater is implemented.
- One Interrupt interface: used to indicate interrupts to the system.
- One Tamper Interrupt interface: used to indicate Tamper interrupts to the system. Only present if LDE.1 or greater is implemented.
- One or more Firewall Configuration interfaces: used for communication between the Firewall Components.
- One or more Protection Size interfaces: one is implemented per each Firewall Component which implements PE.2.
- One Bypass interface per each Firewall Component which implements PE.1 or greater: used to bypass the checks performed by the Protection Extension.

This section contains the following subsections:

C.2.1 Bus Slave and Bus Master interfaces

The number and type of bus protocols for the Bus Slave and Bus Master interfaces is IMPLEMENTATION DEFINED.

Note

When referring to both the Bus Slave interface and Bus Master interface, this manual uses the term *Bus interfaces*.

The number and type of bus protocols must meet the requirements described in *C.1.3 Bus protocol* on page Appx-C-287.

The Bus Slave and Master interfaces of the Firewall, are associated with a Firewall Component within the Firewall. Each Firewall Component has at least one Bus Slave and one Bus Master interface and implements an interconnect between its Bus Slave and Master interfaces. A Firewall is allowed to connect the Bus Master interface of one Firewall Component to the Bus Slave interface of another,

allowing a Firewall to stack Firewall Components. The topology of the interconnect implemented by the Firewall and the Firewall Components, is IMPLEMENTATION DEFINED.

Bus Address Width

When a Firewall Component implements PE.0 there are no requirements on the address widths of the Bus Slave or Master interfaces. This is because all transactions are considered not to pass through the Firewall Component even if, for integration reasons, the bus fabric routes both transaction requests and responses through the Firewall Component.

When PE.1 or greater is implemented there are requirements on the address width of the bus interfaces. *C.4.1 Protection Size and bus address widths* on page Appx-C-305 describes these requirements.

StreamID

This manual requires that a Firewall Component either passes the StreamID (MasterID) unmodified through the Firewall Component, or issues the StreamID (Fixed MasterID) when the transaction is issued.

This manual requires that all Bus interfaces of a Firewall Component follow these rules for StreamID width:

- All Bus Master interfaces must have a MasterID field width greater than or equal to the maximum width of all Bus Slave interfaces, of the Firewall Component.
- When a Bus Slave interface has a MasterID field width less than the maximum Bus Slave interface width, the MasterID field is zero extended.
- When a Bus Master interface has a MasterID field width greater than the maximum Bus Slave interface width the MasterID field is zero extended.

Software can discover the MasterID width for the Firewall Component using the FC_CFG2.MST_ID_WIDTH field. As a Firewall Component can support different Bus interfaces with different-sized MasterID fields, the value in the FC_CFG2.MST_ID_WIDTH is the maximum size of all the Bus interfaces.

Bus Slave incoming transaction properties rules

Each transaction a Firewall Component receives has properties depending on the bus protocol used.

The bus protocol and the Firewall Component can support different properties. The table below shows the value used by the Firewall Component when it implements a property not supported by the Bus Slave interface.

Table C-5 Default bus property values, when supported by Firewall Component, but not by the bus protocol

Property	Firewall Component support	Value when not supported by bus protocol
Instruction	FC_CFG1.INST_SPT is 1	Data
Privilege	FC_CFG1.PRIV_SPT is 1	Unprivileged
Shareability	FC_CFG1.SH_SPT is 1	Non-shareable
Memory Attribute	FC_CFG1.MA_SPT is 1	Normal-iWB_RWA_nT-oWB_RWA_nT
Security	FC_CFG1.SEC_SPT is 1	Non-secure

Note

These values are then used in a future transaction translation. If the bus protocol supports a property not supported by the Firewall Component, the Firewall Component ignores the value.

Arm strongly recommends that the transaction properties supported by the Firewall Component are equal to or greater than those supported by the bus protocol used by the Bus Slave interfaces of the Firewall Component.

If the bus protocol for a transaction does not support one of these properties and a value other than those listed in the table above is required, then the Firewall Component must:

- Be configured with support for the property
- Bus slave interface signal for the property is tied to the desired value

Bus Master output transactions properties rules

A Firewall Component must not issue an outgoing transaction which has illegal properties.This is true, whether or not it is because of illegal incoming properties or bad software programming when translation is being applied.

When outputting the transaction, the following rules must always be obeyed when the Firewall Component implements PE.1 or greater:

- If a transaction has a memory attribute of Device or Normal Inner Non-Cacheable Outer Non-Cacheable the output shareability must be Outer Shareable.
- If a transaction has a memory attribute of Normal with either the Inner or Outer cache level being Non-cacheable, the transaction is considered to be read-no-allocate write-no-allocate and non-transient at that level regardless of the programmed or incoming values. A non-cacheable access is considered read-no-allocate write-no-allocate and non-transient. This is independent of programmed or incoming value.
- If a transaction has a memory attribute of Normal with a cacheable type that is read-no-allocate and write-no-allocate, the memory has no transient attribute and is always considered non-transient independent of the programmed or incoming value.

The above requirements apply independently of:

- The level of TE implemented
- The value of the property's translation enable

For more information on memory and shareability attributes see *Output transaction memory attribute property* on page Appx-C-357 and *Output transaction memory attribute property* on page Appx-C-357.

When PE.0 is implemented transactions are always considered to be issued without going through the Firewall Component even, if for integration reasons, the bus fabric always routes both transaction requests and responses through the Firewall Component.

When the Firewall Components support a property which the protocol for the Bus Master interface does not support, the property is omitted when the transaction is issued on the interface.

When the bus protocol of the Bus Slave interface supports a property which the Firewall Component does not support, the Firewall Component ignores the property. Any output transactions have the property:

- Unmodified, if support by the protocol of the Bus Master interface
- Omitted, if not support by the protocol of the Bus Master interface

Independent of which option is selected, the above rules must always be followed for transaction output by Firewall Components implementing PE.1 or greater.

Arm strongly recommends that the transaction properties supported by the Firewall Component are equal to or greater than those supported by the bus protocol used by the Bus Master interfaces of the Firewall Component.

C.2.2 Programming interface

The Programming Interface must be present on the Firewall Controller and is used to perform configuration accesses to the configuration of the Firewall.

The Programming interface must adhere to the requirements described in section *C.4.4 Transaction processing* on page Appx-C-316.

Arm strongly recommends that the Firewall Controller and the bus protocol used for the Programming interface supports the following transactions properties:

- Instruction
- Security
- Privilege

The Programming interface can be combined with one of Bus Slave interfaces of the Firewall Controller, provided the following are met:

- All the requirements for both interfaces are still met.
- The Bus Slave interface is not capable of preventing a transaction for the Programming interface making progress.
- The Programming interface and the Bus Slave interface support the same bus protocol and level of properties.

C.2.3 Power Control interfaces

The Power Control interface must be able to:

- Indicate the current power mode of the power domain, where the Firewall Component resides
- Minimally support the ability to indicate whether the Firewall Component is operational or not. The Firewall Component is considered operational when it is guaranteed not to lose any architectural state. It is considered non-operational when there is a possibility of a full or partial loss of architectural state could occur.
- Provide a mechanism for the Firewall Component to indicate its desired power mode
- Provide a mechanism for the Firewall Component to accept or deny a change in power mode

Arm strongly recommends that the Power Control interface is either a P or Q-Channel interface and follows the guidelines in the *Arm® Power Control System Architecture Version 1.0 Architecture Specification.*

An implementation of the Firewall is required to have a Power Control interface per hardware entity which makes up the Firewall.

The power modes, supported by the Firewall Component, and whether the Firewall Component is considered operational or non-operational in that power mode, is IMPLEMENTATION DEFINED.

Some power modes, which a Firewall Component can enter do not cause the loss of any architectural state, but the Firewall Component is unable to process transactions on the Bus Slave or Configuration interfaces. In these power modes the Firewall Component is still considered operational, as no state is loss. It must be able to exit this power mode and enter one where it is able to process new transactions on the Bus Slave or Configuration interfaces when they arrive. The method by which this is achieved is IMPLEMENTATION DEFINED but, must be achieved without any software interaction.

Note

Depending on the power control of the system which the Firewall is integrated into a limited amount of software interaction may be required with the power controller and is outside the scope of this specification.

It is allowed for a Firewall Component to enter the Disconnected state and not to lose any architectural state. This could be because the power controller did not complete the transition to a power mode where state would be lost.

For example, an implementation can include support for ON, OFF and FULL_RET power modes. This allows for the Firewall Component to enter a retention state when there are no transactions to process on any of the Bus Slave or Firewall Configuration interfaces. In this case, ON and FULL_RET are

considered operational modes as no architectural state is loss. Requirements are placed on the system integration to cause an exit from the FULL_RET power mode when either:

- A transaction arrives to be processed, on any Bus interface.
- An access arrives on the Firewall Configuration interface.

Note

Depending on the level of PE and ME implemented by the Firewall Component, processing the transaction refers to performing the protection checks, detection of errors or both.

Before a Firewall Component enters a power mode, which causes it to transition from being operational to non-operational or vice versa, the Firewall Component must perform the required Firewall Component state transition. For more information on Firewall Component states, see *Firewall Component Status (PE_ST)* on page Appx-C-322.

If the SRE.1 is implemented, it is IMPLEMENTATION DEFINED whether the Firewall supports:

- Placing the shadow registers into retention when there are no accesses to the shadow registers
- Firewall Controller is in a power mode, where it is considered non-operational, but retaining the contents of the shadow registers

Arm strongly advises that the Power Control interface, for the Firewall Controller, supports placing the shadow registers into retention when there are no accesses to the shadow registers.

In implementations which support placing the Firewall Controller in a non-operational mode, the FW_SR_CTRL.SR_PWR determines whether the shadow registers are required to retain their values or not. For more information, see *Firewall Component Status (PE_ST)* on page Appx-C-322.

It is IMPLEMENTATION DEFINED, how the Firewall Controller knows the shadow register contain valid data after it leaves the Disconnected state.

C.2.4 Clock Control interfaces

This section describes Clock Control interfaces.

The Clock Control interface must be capable of:

- Indicating whether the clock, of the clock domain, is gated
- Provide a mechanism for the Firewall Component to indicate its desire for the clock to be gated or not
- Provide a mechanism for the Firewall Component to accept or deny the clock being gated

Arm strongly recommends that the Clock Control interface is implement as Q-Channel interface and follows the guidelines in the *Arm® Power Control System Architecture Version 1.0 Architecture Specification.*

An implementation of the Firewall must have at least one Clock Control interface per each clock used by the Firewall. It is IMPLEMENTATION DEFINED when multiple Firewall Components sharing the same clock, share a Clock Control interface. If the Clock Control interface is shared it is IMPLEMENTATION DEFINED how the Firewall Components are informed about the changes in the clock state.

The clock can only be gated if the Firewall Component accepts the clock gating request made on the Clock Control interface. When the clock is gated, the Firewall Component is not required to process any transactions on the Bus Slave or Configuration interfaces of the Firewall Component. The Firewall Component is required to request a clock when it has:

- Transactions outstanding on the Bus Slave or Configuration interfaces
- A request on the power control interface

Whether the clock is gated or not has no effect on anything other than the Firewall Component.

——— **Note** ———

This mechanism is in addition to any internal clock gating that may be implemented within the Firewall Component.

———————————

C.2.5 Lockdown interfaces

This section describes Lockdown interface.

The lockdown interface is an optional interface and is only implemented if the Firewall implements LDE.1 or greater. The lockdown interface prevents Firewall Components from changing their lockdown state, under certain conditions, when asserted high.

——— **Note** ———

For more information on the conditions where the Firewall Component's lockdown state can change see *C.11.3 Lockdown Extension* on page Appx-C-378.

———————————

For more information about the interaction between the SE and LDE see *C.8 Security Extension* on page Appx-C-365 and *C.11.3 Lockdown Extension* on page Appx-C-378.

The Lockdown interface is only implemented on the Firewall Controller.

C.2.6 Interrupt interface

This section describes Interrupt interface.

The Interrupt interface provides a method for the Firewall to generate interrupts to the system. Interrupts generated by the Firewall, use a wired interrupt. The Interrupt interface has the following requirements:

- Must be able to indicate when an interrupt has been generated
- Indicates when the interrupt has been cleared
- Remains asserted while any interrupt status field, it is associated with, is set to 1
- It must not be possible to receive the interrupt, without being able to observe the required registers have been updated. For more information see *C.11.6 Interrupts* on page Appx-C-381 on the requirements for each interrupt type.

C.2.7 Tamper Interrupt interface

This section describes Tamper Interrupt interface.

The Tamper Interrupt interface provides a method for the Firewall to generate tamper interrupts to the system. The Tamper Interrupt interface is an optional interface which is only implemented when LDE.1 or greater is implemented. This interface must be separate to the Interrupt interface else it be routed to another interrupt controller in the system. For example, the root of trust in the system. The Tamper Interrupt interface is similar to the Interrupt interface and must use wired interrupts. The Tamper Interrupt interface must follow these requirements:

- Must be able to indicate when an interrupt has been generated
- Indicates when the interrupt has been cleared
- Remains asserted while either TR_VLD or TMP_ST_OVERFLW fields, in the FW_TMP_CTRL register, are set to 1
- It must not be possible to receive the interrupt, without being able to observe that the required registers have been updated. For more information see *C.11.6 Interrupts* on page Appx-C-381 on the requirements for each interrupt type.

C.2.8 Firewall Configuration interface

This section describes the Configuration interface.

The Firewall Configuration interface is private to the Firewall, however depending on how the Firewall is implemented the interface may be external or internal. Independent of whether the interface is external or internal to the Firewall, it must meet the following requirements:

- Bi-directional communication between the Firewall Controller and the other Firewall Component
- Capable of performing the handshakes and request
- Capable of performing configuration accesses, received by the Firewall on the Programming interface to the configuration registers of the Firewall Components. See section *C.2.2 Programming interface* on page Appx-C-291 for more information on the Programming interface.
- Private to the Firewall. It must only be accessible by Firewall Components, even if these components would not be able to access any Firewall registers or observe any handshakes or requests sent over the interface.

Note

It is not required that a Firewall Component can communicate with another Firewall Component which is not the Firewall Controller.

C.2.9 Protection Size interface

This section describes Firewall Protection Size interface.

Each Firewall Component that implements PE.2 within a Firewall, must have a corresponding Protection Size interface.

The Protection Size Interface is an 8-bit signal, which allows setting the Protection Size of the Firewall Component. The Protection Size can be set, in power of twos, between the Minimum Region Size (MNRS) and the Maximum Region Size (MXRS). See *C.4.1 Protection Size and bus address widths* on page Appx-C-305 for more information on the MNRS, MXRS and Protection Size. The Protection Size can also be set to 0 to force all transactions to be treated as failing the checks irrespective of the transaction and the regions defined in the Firewall Component. The encoding of the Protection Size interface is the same as the FC_CFG2.PROT_SIZE field. See *Firewall Component Configuration Register 2 (FC_CFG2)* on page Appx-C-301 for more information of the encoding of FC_CFG2.PROT_SIZE.

The Protection Size interfaces are always present on the Firewall Controller. The signal is sampled when the Firewall Controller enters the Connected state and the shadow registers do not contain any valid values.

Note

When SRE.0 is implemented there are no shadow registers. Therefore, the Firewall Controller always samples the Protection Size interface on entry into the Connected state.

In Appendix C, when referring to the Protection Size interface value, it is referring to the sampled value.

As part of the Connect handshake, the sampled value of the Protection Size Interface must be communicated to the Firewall Component.

Once the Protection Size Interface is sampled the values of the following fields must be updated to reflect the sampled value:

- FC_CFG2.PROT_SIZE
- RGN_SIZE.SIZE for region 0 of a Firewall Component implementing PE.2

After the Protection Size Interface is sampled, any change in the value has no effect on the behavior of the Firewall Component until the interface is next sampled.

C.2.10 Bypass interface

This section describes the Bypass interface.

There is a Bypass interface per Firewall Component which implements PE.1 or greater and controls whether the Firewall Component is bypassed or not. The value of the Bypass interface of the Bypass interface of the Firewall Component can be read by software using the PE_BPS.BYPASS_IF_ST field.

C.3 Firewall Component

Overview of Firewall component.

This section contains the following subsections:

C.3.1 Overview

A Firewall Component can be configured to provide protection, monitoring or both depending on the level of PE and ME implemented by the Firewall Component.

This section describes the common features that all Firewall Components provide. Sections *C.4 Protection Extension* on page Appx-C-305 to *C.10 Save and Restore Extension* on page Appx-C-370 provide descriptions of each extension. Section *C.11 Firewall Controller* on page Appx-C-373 provides a description of the differences between the Firewall Controller and a standard Firewall Component.

Each Firewall Component has 64KB allocated from the Firewall memory map for its own configuration registers, but only the first 4KB is used and the remaining 60KB is Reserved and generates a Configuration Access Error if accessed. In the following sections, the configuration registers of a Firewall Component are described, alongside the offset address within the allocated space. For a full view of all the configuration registers of a Firewall Component or the memory map of the Firewall, see section *C.13 Programmers model overview* on page Appx-C-398. Any unused address space within the 4KB is Reserved and generates a Configuration Access Error if accessed.

C.3.2 Common registers

This section describes the Firewall common registers.

There are some common registers which all Firewall Components must implement. These are related to:

- The capabilities of the Firewall and individual Firewall Components
- The configuration of the Firewall and individual Firewall Components

Capability registers

A Firewall Component has the Firewall Component Capability registers, FC_CAP{0-3}.

Software can use these registers to find out the extension levels the Firewall and individual Firewall Component use.

Table C-6 Summary of Capability registers

Offset	Short Name	Access	Name
0xFA0	*FC_CAP0* on page Appx-C-297	RO	Firewall Component Capability Register 0
0xFA4	*FC_CAP1* on page Appx-C-299	RO	Firewall Component Capability Register 1
0xFA8	*FC_CAP2* on page Appx-C-299	RO	Firewall Component Capability Register 2
0xFAC	*FC_CAP3* on page Appx-C-299	RO	Firewall Component Capability Register 3

Firewall Component Capability Register 0 (FC_CAP0)

The following table gives a bit-level description of Firewall Component Capability Register 0.

Table C-7 Register 0 (FC_CAP0)

Bits	Name	Description	Type	Reset
[31:28]	-	Reserved	RO	`0x0`
[27:24]	SE_LVL	Level of the Security Extension used by the Firewall. `0x0`: SE.0 is implemented `0x1`: SE.1 is implemented All other values are Reserved. For Firewall Components, other than 0, this field is Reserved and treated as RAZ/WI.	RO	IMPL_DEF
[23:20]	SRE_LVL	Level of the Save and Restore Extension used by the Firewall. `0x0`: SRE.0 is implemented `0x1`: SRE.1 is implemented All other values are Reserved. For Firewall Components, other than 0, this field is Reserved and treated as RAZ/WI.	RO	IMPL_DEF
[19:16]	LDE_LVL	Level of the Lockdown Extension implemented by the Firewall. `0x0`: LDE.0 is implemented `0x1`: LDE.1 is implemented `0x2`: LDE.2 is implemented All other values are Reserved. For Firewall Components, other than 0, this field is Reserved and treated as RAZ/WI.	RO	IMPL_DEF
[15:12]	TE_LVL	Level of the Translation Extension implemented by Firewall Component 0. `0x0`: TE.0 is implemented `0x1`: TE.1 is implemented `0x2`: TE.2 is implemented All other values are Reserved. This field must always be `0x0` when the FC_CAP0.PE_LVL is `0x0`.	RO	IMPL_DEF
[11:8]	RSE_LVL	Level of the Region Size Extension implemented by Firewall Component. `0x0`: RSE.0 is implemented `0x1`: RSE.1 is implemented All other values are Reserved. This field must always be `0x0` when the FC_CAP0.PE_LVL is `0x0`.	RO	IMPL_DEF

Table C-7 Register 0 (FC_CAP0) (continued)

Bits	Name	Description	Type	Reset
[7:4]	ME_LVL	Level of the Monitor Extension implemented by the Firewall Component. `0x0`: ME.0 is implemented `0x1`: ME.1 is implemented `0x2`: ME.2 is implemented All other values are Reserved.	RO	IMPL_DEF
[3:0]	PE_LVL	Level of the Protection Extension implemented by the Firewall Component. `0x0`: PE.0 is implemented `0x1`: PE.1 is implemented `0x2`: PE.2 is implemented All other values are Reserved.	RO	IMPL_DEF

The values of FC_CAP0.{PE_LVL,ME_LVL,RSE_LVL,TE_LVL} can be different for each Firewall Component within the Firewall.

Firewall Component Capability Register 1(FC_CAP1)

The following table gives a bit-level description of Firewall Component Capability Register 1.

Table C-8 Register 1(FC_CAP1)

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	`0x0000_0000`

Firewall Component Capability Register 2 (FC_CAP2)

The following table gives a bit-level description of the Firewall Component Capability Register 2.

Table C-9 Register 2 (FC_CAP2)

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	`0x0000_0000`

Firewall Component Capability Register 3 (FC_CAP3)

The following table gives a bit-level description of Firewall Component Capability Register 3.

Table C-10 Register 3 (FC_CAP3)

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	`0x0000_0000`

Configuration registers

A Firewall Component has the Firewall Component Configuration registers, FC_CFG{0-3}.

Software can use these registers to find out how the Firewall Component has been configured, for example, the address width of the Bus interfaces.

Table C-11 Configuration registers FC_CFG{0-3}

Offset	Short Name	Access	Name
0xFB0	*FC_CFG0* on page Appx-C-300	RO	Firewall Component Configuration Register 0
0xFB4	*FC_CFG1* on page Appx-C-300	RO	Firewall Component Configuration Register 1
0xFB8	*FC_CFG2* on page Appx-C-301	RO	Firewall Component Configuration Register 2
0xFBC	*FC_CFG3* on page Appx-C-303	RO	Firewall Component Configuration Register 3

Firewall Component Configuration Register 0 (FC_CFG0)

The following table gives a bit-level description of Firewall Component Configuration Register 0.

Table C-12 Register 0 (FC_CFG0)

Bits	Name	Description	Type	Reset
[31:5]	-	Reserved	RO	0x000_0000
[4:0]	FC_ID	Firewall Component ID	RO	IMPL_DEF

Firewall Component Configuration Register 1 (FC_CFG1)

The following table gives a bit-level description of Firewall Component Configuration Register 1.

Table C-13 Register 1 (FC_CFG1)

Bits	Name	Description	Type	Reset
[31:21	-	Reserved	RO	0x000
[20]	SEC_SPT	Firewall Component support for checking the security of the incoming transaction and setting the security of the outgoing transaction. 0b0: Not supported 0b1: Supported This field is 0b0 when SE.0 is implemented and 1 when SE.1 is implemented.	RO	IMPL_DEF
[19]	MA_SPT	Firewall Component support for setting the memory type of the outgoing transaction. 0b0: Not supported 0b1: Supported This field is 0b0 when TE.0 is implemented.	RO	IMPL_DEF
[18]	SH_SPT	Firewall Component support for setting the shareability of the outgoing transaction. 0b0: Not supported 0b1: Supported This field is 0b0 when TE.0 is implemented.	RO	IMPL_DEF
[17]	INST_SPT	Firewall Component support for checking whether the incoming transaction is an instruction or data access and setting the instruction property of the outgoing transaction. 0b0: Not supported 0b1: Supported	RO	IMPL_DEF

Table C-13 Register 1 (FC_CFG1) (continued)

Bits	Name	Description	Type	Reset
[16]	PRIV_SPT	Firewall Component support for checking the privileged level of the incoming transaction and setting the privileged level of the outgoing transaction. `0b0`: Not supported `0b1`: Supported	RO	IMPL_DEF
[15:14]	-	Reserved	RO	`0x00`
[13:12]	NUM_MPE	Number of MPEs implemented, per region, in the Firewall Component. `0b00`: 1 MPE per region `0b01`: 2 MPEs per region `0b10`: 3 MPEs per region `0b11`: 4 MPEs per region This field is Reserved and treated as RAZ/WI when PE.0 is implemented.	RO	IMPL_DEF
[11]	-	Reserved	RO	`0x0`
[10:8]	MNRS	Minimum Region Size. `0x1`: 64B `0x2`: 128B `0x3`: 256B `0x4`: 512B `0x5`: 1KB `0x6`: 2KB `0x7`: 4KB This field is Reserved and treated as RAZ/WI when PE.0 is implemented.	RO	IMPL_DEF
[7:0]	NUM_RGN	Number of regions implemented in the Firewall Component. The number of regions implemented is NUM_RGN+1. This field is Reserved and treated as RAZ/WI when PE.0 is implemented.	RO	IMPL_DEF

Firewall Component Configuration Register 2 (FC_CFG2)

The following table gives a bit-level description of Firewall Component Configuration Register 2.

Table C-14 Register 2 (FC_CFG2)

Bits	Name	Description	Type	Reset
[31]	SINGLE_MST	Whether the Firewall Component supports a single MasterID. `0b0`: Firewall Component supports more than one MasterID. `0b1`: Firewall Component supports only one MasterID.	RO	IMPL_DEF
[30:21]	-	Reserved	RO	`0x0_0000`

Table C-14 Register 2 (FC_CFG2) (continued)

Bits	Name	Description	Type	Reset
[20:13]	PROT_SIZE	Protection Size The value of this field indicates the range of addresses which the Firewall Component protects. `0x00`: 0B `0x05`: 32B `0x06`: 64B … `0x0C`: 4K `0x0D`: 8KB … `0x40`: 16EB The value of this field depends on the level of Protection Extension implemented by the Firewall Component. PE.0: `0x00` PE.1: Same value as FC_CFG2.MXRS. PE.2: Sampled value of the Protection Size interface for the Firewall Component.	RO	IMPL_DEF
[12:8]	MST_ID_WIDTH	Maximum MasterID field width of the Bus interfaces of the Firewall Component. `0x00`: 1 bit of MasterID `0x01`: 2 bits of MasterID … `0x1F`: 32 bits of MasterID When a Firewall Component supports more than one Bus Slave or Master interface this field is set to the largest width used.	RO	IMPL_DEF
[7]	-	Reserved	RO	`0x0`
[6:0]	MXRS	The Maximum Region Size the Firewall Component Supports. `0x05`: 32B `0x06`: 64B … `0x0C`: 4KB `0x0D`: 8KB … `0x40`: 16EB All other values are Reserved. This field is Reserved and treated as RAZ/WI invalid for a Firewall Component which implements PE.0.	RO	IMPL_DEF

Arm recommends that when FC_CFG2.SINGLE_MST reads as 1, that FC_CFG1.NUM_MPE reads as 00, as defining more than one MPE for a region would lead to a programming error as both entries would have the same MasterID value.

Firewall Component Configuration Register 3 (FC_CFG3)

The following table gives a bit-level description of Firewall Component Configuration Register 3.

Table C-15 Register 3 (FC_CFG3)

Bits	Name	Description	Type	Reset
[31:6]	-	Reserved	RO	0x000_0000
[5:0]	NUM_FC	Number of Firewall Components implemented. The number of Firewall Components which are implemented in the Firewall is NUM_FC +1. For Firewall Components, other than 0, this field is Reserved and treated as RAZ/WI.	RO	IMPL_DEF

C.3.3 Firewall Component states

This section describes Firewall Component states.

Each Firewall Component can be in one of the following states:

Disconnected

Firewall Component is unable to process transaction and configuration accesses are dependent on the level of SRE implemented.

- SRE.0: Access to configuration registers of a Firewall component, in the Disconnected state, generate a Configuration Access Error response.
- SRE.1: Access to configuration register of a Firewall Component, in the Disconnected state complete without generating a Configuration Access Error response. Unless the access would cause a Configuration Access Error if the Firewall Component was in the Connected state. Depending on whether the register and fields accessed by the access, are saved and restored, the access may either:
  - For read access return 0s or the value of the field
  - For write access cause an update or be treated as WI

For more information on the behavior of the different registers when SRE.1 is implemented see section *C.10.3 Register Behavior when SRE.1 Implemented* on page Appx-C-370.

Connecting

Firewall Component is transitioning from Disconnected to Connected. For configuration accesses it is IMPLEMENTATION DEFINED, which one of the following the implementation implements:

- Accesses behave as if the Firewall Component, is still in the Disconnected state. Depending on the level of the SRE implemented:
  - SRE.0: Generate a Configuration Access Error.
  - SRE.1: Allow the access. If the access updates a register, which has not been restored by the Firewall Controller, it must not be possible for the restore process to complete and the old value be used.
- Accesses will be stalled, at the Programming interface of the Firewall Controller, whilst the Connect handshake is completed.

——— **Note** ———

Arm recommends that an implementation selects:

- Option 1a when SRE.0 is implemented
- Option 2 when SRE.1 is implemented
- Transactions are not stalled indefinitely to avoid deadlocking the system.

———————————

Connected

Firewall Component is able to process transactions and register accesses complete as normal.

Disconnecting

Firewall Component is transitioning from Connected to Disconnected. Configuration accesses behavior as described for the Connecting state.

A Firewall Component transitions between these states based on the power mode of the domain which the Firewall Component resides in.

C.4 Protection Extension

This section describes the Protection Extension.

Note

In this section, any references to a Firewall Component implicitly implies that the Firewall Component implements PE.1 or greater.

For the masters connected to the Bus Slave interfaces of the Firewall Component, the Protection Extension enables the Firewall Component to compartmentalize the address space of the Bus Master interfaces.

Compartmentalization is achieved using regions. A Firewall Component implements between 1 and 256 regions. Each region defines the access permissions to an address range based on the master which issued the transaction. Using the Region Window Entry (RWE) registers, software can program a region. For more information on regions and Region Window Entry see sections *C.4.2 Regions* on page Appx-C-306 and *C.4.7 Registers* on page Appx-C-319 respectively.

Alongside the regions, a Firewall Component which implements PE.1 or greater, also implements at least one fault entry. Each fault entry provides details about a transaction which has faulted. Software accesses the individual fault entries using the Fault Window Entry (FWE). For more information on fault entries and Fault Window Entry, see sections *C.4.3 Fault entries* on page Appx-C-313 *and* on page Appx-C-306*C.4.7 Registers* on page Appx-C-319 respectively.

When in the Connected state, a Firewall Component processes transactions as described in section *C.4.4 Transaction processing* on page Appx-C-316. When processing, the Firewall Component performs checks on the transactions. Transactions which pass proceed to their destination are referred to as an allowed transactions. For transactions which fail, also referred to as faulting transactions, the Firewall Component blocks the transaction from continuing to its destination. This is referred to as terminating the transaction.

This section contains the following subsections:

C.4.1 Protection Size and bus address widths

This section gives the rules of Protection Size and bus address width.

A Firewall Component, which supports PE.1 or greater, has the following properties:

- Minimum Region Size, MNRS: smallest region size which can be defined
- Maximum Region Size, MXRS: largest region size which can be defined
- Protection Size, PROT_SIZE: the range of address bits which the Firewall Component uses to match the incoming transaction against a region when performing address translation. This value represents the bits which can be modified.
- Bus Slave interface address width, BSAW, per Bus Slave interface implemented
- Bus Master interface address width, BMAW, per Bus Master interface implemented

The following rules apply:

- MXRS must always be >= MNRS
- MNRS must be one of the following: 32B, 64B, 128B, 256B, 512B, 1KB, 2KB, 4KB
- MXRS must be between 32B and 16EB in powers of 2
- Protection Size must be between MNRS and MXRS inclusive, or 0:
  - When PE.1 is implemented Protection Size is always equal to MXRS.
  - When PE.2 is implemented Protection Size is set by the sampled value of the Protection Size interface:
    - If the Protection Size is set to a value less than MNRS, except for 0, it is treated as if it was set to the MNRS value.
    - If the Protection Size is set to a value greater than MXRS, it is treated as if it was set to the MXRS value.
    - When the Protection Size is set to 0, all transactions are treated as failing the checks when the Firewall Component is not bypassed.
- Max(BSAW) must never be greater than Min(BMAW).
- A Firewall Component treats all incoming transactions as having an address width of 64 bits:
  - When a Bus Slave interface address width is less than 64 bits the address is zero extended to 64 bits.
  - When a Bus Master interface address width is less than 64 bits the address is truncated to match the width of the Bus Master interface.
- When a transaction is received on a Bus Slave interface there are the following requirements on the address bits of the incoming transactions:
  - Address bits log2(MXRS) to Max(BSAW)-1, if BSAW is greater, are ignored.

    Note

    This can lead to aliasing if Firewall Component's Max(BSAW) is greater than log2(MXRS). It is the responsibility of the integrator of the Firewall to avoid alias if it can occur. It is only considered aliasing if the bits of the address greater than the log2(MXRS)-1 can take more than a single value. If, due to an upstream component, the address bits of all transactions above this are always the same it is not considered aliasing.

  - Address bits PROT_SIZE to log2(MXRS)-1 must be 0, otherwise the transaction is automatically considered to fail the protection checks. Depending on the value of the PE_ST.FLT_CFG the transaction is either:
    - Terminated without a fault entry
    - Terminated with a fault entry. If a fault entry is generated, then a transaction fault is generated.

  Note

  When the Firewall Component is bypassed this requirement no longer applies.

- When a transaction is to be issued on a Bus Master interface address bits log2(MXRS) to Max(BMAW)-1, if BMAW is greater, are passed through unaltered.

C.4.2 Regions

This section describes Firewall regions.

A Firewall Component has between 1 and 256 regions. The number of regions implemented in a Firewall Component is set in FC_CFG1.NUM_RGN. A region has properties which software can program using the *Region Window Entry* (RWE). For more information on programming a region, see section *C.4.7 Registers* on page Appx-C-319.

Region properties

A region defines an area of memory which certain masters can perform certain memory operations on.

A region can have the following properties:

- Base address
- Upper address
- Size
- 1-4 *Master Permission Entries* (MPE), each containing:
  - MasterID
  - Enables for combinations of Secure and Non-secure, privileged and unprivileged, read, write and execute accesses.

Not all region properties are programmable by software. Some of the properties are dependent on:

- The level of extensions implemented
- The design time configuration of Firewall Component
- How software has programmed the Firewall Component

Region properties on page Appx-C-306 to *Master Permission Entries (MPEs)* on page Appx-C-309 describes each property in detail.

Base and upper address

This section describes the base and upper address of a region.

The base address property of a region sets the lower address boundary that the region will match on incoming transactions. The base address is considered inclusive to the region, i.e. the address of the incoming transaction must be greater than or equal to the base address.

The upper address property of a region sets the upper address boundary that the region will match incoming transactions on. The upper address is considered inclusive to the region, i.e. the address of the incoming transaction must be less than or equal to the upper address.

The base and upper address properties can be either:

- Fixed: the value of the property is defined at design time.
- Directly programmable: the value of the property can be programmed by software by writing to the field.
- Indirectly programmable: the value of the property can be programmed by software setting the size property. This only applies to the upper address, which becomes equal to the base address plus the size minus 1.

The table below shows the behavior of the property against the various levels of extension, configuration and programming options.

Table C-16 Behavior of a region's base and upper address

Property	Level of RSE	Level of PE	RGN_SIZE.MULnPO2	Behavior
Base	X	1	X	Fixed
	X	2	X	Directly programmable
Upper	X	1	X	Fixed
	0	2	X	Indirectly programmable, via the size property
	1	2	0	Indirectly programmable, via the size property
	1	2	1	Directly programmable

In all cases:

- Both the base and upper address must be an integer multiple of the MNRS. An attempt to program a value which is not an integer multiple of MNRS is round down to the nearest integer multiple of MNRS to the value.
- The base address must be within the Protection Size of the Firewall Component. If the base address is set to a value greater than the Protection Size, no transactions will match against the region.

Note

Arm recommends that the upper address is never programmed to be less than the base address. If a region is programmed like this, then no transaction will match against the region as it will always fail the check. The base and upper address may be set to the same value to request a region which is equal to the MNRS in size starting at the base address.

The bits of the base and upper address used when matching an incoming transaction to the region depend on the configuration of the region:

- When either RSE.0 is or RSE.1 is implemented and MULnPO2 is 0, the Firewall Component treats the base address as being aligned to the size of the region.
- When RSE.1 is implemented and MULnPO2 is 1, the Firewall Component treats the base and upper addresses as being aligned to MNRS of the Firewall Component.

Note

Arm strongly recommends that software always sets the base and upper address values to match the behavior of the Firewall Component.

For more information on how the base and upper addresses are used in region matching see section *Region matching* on page Appx-C-311.

Size

This section describes region size.

A region has a size in bytes associated with it. The size property of a region can be either fixed or programmable. The range of values size can be set to, either via software or at design time, is shown in the table below.

Table C-17 Behavior and legal values of a region's size property

Level of PE	Legal values for size	Behavior
1	MNRS to MXRS inclusive and 0	Fixed
2	MNRS to MXRS inclusive and 0	Programmable

The size of a region can be defined as 0 to prevent any transaction matching against the region.

If a regions size is:

- Less than MNRS: it is treated as if the size was set to 0
- Greater than MXRS: it is treated as if the size was set to 0

For more information on MNRS and MXRS, see section *C.4.1 Protection Size and bus address widths* on page Appx-C-305.

For example, in a Firewall Component with:

- MNRS of 4KB
- MXRS of 1MB

The table below shows whether certain values of region size are legal or not.

Table C-18 Region Size legality example

Region size	MULnPO2	Legal	Notes
32B	X	No	Size is below MNRS
4KB	X	Yes	Size is between the MNRS and MXRS and a power of 2
8KB	X	Yes	Size is between the MNRS and MXRS and a power of 2
12KB	0	No	Region is defined as a power of 2
	1	Yes	Region is defined with a base and upper address. Both must be an integer multiple of MNRS.
1GB	X	No	Size is above MXRS

Master Permission Entries (MPEs)

This section describes MPEs of a region.

There are up to four MPEs, MPE{0-3}, per each region which allow up to four different permissions to be defined. Each MPE defines the access permission for a specific master or, in the case of MPE0, for any master to that region. The MPE is only used if:

- The MasterID of the transaction and the MPE match.
- The MPE is configured to match any MasterID, only for MPE0.

The number of MPEs implemented, per region by a Firewall Component, can be discovered using the FC_CFG2.NUM_MPE field.

Note

If more than four masters must have access to a single area of memory, then software should define more regions with the same base address, size or upper address and use the MPE entries of the additional regions. It is valid to define the same area of memory in multiple regions so long as the same master is not defined in more than one MPE for the same area of memory. The Firewall Component will generate a programming fault if a transaction matches more than one region and MPE pair.

Note

Arm recommends that if software programs MPE0 to match against any MasterID that:

- No other MPEs are enabled for that region.
- No other regions, for the same address range are defined.

Otherwise a programming fault is generated.

Each MPE is made up of:

- MasterID, to be matched against the incoming transaction
- *Master Permissions List* (MPL)

The ranges of permissions provided, depends on the levels of extensions implemented and the Firewall Component support for transaction properties. The table below shows the permissions.

Table C-19 Master permissions supported by Firewall Component

Permission	Description
NSUR	Non-secure unprivileged read
NSUW	Non-secure unprivileged write
NSUX	Non-secure unprivileged execute

Table C-19 Master permissions supported by Firewall Component (continued)

Permission	Description
NSPR	Non-secure privileged read
NSPW	Non-secure privileged write
NSPX	Non-secure privileged execute
SUR	Secure unprivileged read
SUW	Secure unprivileged write
SUX	Secure unprivileged execute
SPR	Secure privileged read
SPW	Secure privileged write
SPX	Secure privileged execute

Each permission type allows the defined master to perform access of that type when set to 1. For example, a non-secure privileged read transaction from Master 0 when NSPR is set to 1, will pass the checks, but a Secure unprivileged write from Master 0 when SUW is set to 0 will fail the checks.

Both the permissions and the MasterID are ignored unless the enable bit for the MPE is set to 1.

Enables

This section summarizes region enables.

Region enable

The region enable is always implemented. It is used to control whether incoming transactions can match against this region.

When the region enable is set to 1, then any write to modify the following region properties generates a Configuration Access Error:

- Base address
- Size
- Upper address
- Translation address
- Translation properties

MPE enable

There is a Master Permission Enable per MPE which is implemented for the region.

This enable controls whether a transaction can match against the MPE for the region. When the MPE enable is set to 1, any write to modify the associated MPE is ignored.

Region lock

The region lock is implemented when the LDE.2 is implemented.

When the region lock is set to 1 any attempt to change any property of the region generates:

- A Configuration Access Error
- A Tamper interrupt and Tamper report, if there is no valid Tamper report present. Otherwise, a Tamper Overflow interrupt is generated.

The behavior of an attempt to update the region lock depends on the value of the region lock and the lockdown state of the Firewall Component, when it is set to 1, depends on the lockdown state of the Firewall Component:

- Open: Write is allowed
- Partial and region is locked generates:
  - A Configuration Access Error
  - A Tamper interrupt and Tamper report, if there is no valid Tamper report present. Otherwise, a Tamper Overflow interrupt is generated.
- Full: Generates:
  - A Configuration Access Error
  - A Tamper interrupt and Tamper report, if there is no valid Tamper report present. Otherwise, a Tamper Overflow interrupt is generated.

For more information on LDE see section *C.11.3 Lockdown Extension* on page Appx-C-378.

Default regions

This section describes default regions.

When PE.2 is implemented, the Firewall Component implements a default region (region 0), which behaves differently to the other regions within the Firewall Component. The default region has the following changes to the other regions:

- Base address is fixed and set to a value of all 0s.
- Size is fixed and set equal to the FC_CFG2.PROT_SIZE field.
- RGN_TCFG2.ADDR_TRANS_EN is 0 and treated as RAZ/WI.
- RGN_SIZE.MULnPO2 is 0 and treated as RAZ/WI.

Note

When PE.2 is implemented and FC_CFG1.NUM_RGN is 0, then only the default region is implemented.

Default region matching

This section describes default region matching.

A transaction only matches against the default region when the default region is enabled, and the transaction does not match any other region.

If a transaction matches a region, but fails to find any MPE with a MasterID matching the transaction, or MPE0 is not configured to match any MasterID, it can then be checked against the default region. Transactions do not match against the default region, if they match another region, but fail the permission check for:

- An MPE which has the MasterID matching the transaction
- MPE0 of the region is configured to match any MasterID

If a transaction matches more than one MPE in the default region, then a Programming Error is generated for the transaction.

Region matching

This section describes region matching.

The Firewall Component performs checks referred to as protection logic checks, or checks. This section describes the process for performing these checks. These checks are only performed if the Firewall Component's protection logic is enabled and not bypassed. For more information on the behavior of a Firewall Component and bypass see section *C.4.6 Bypass interface* on page Appx-C-319.

When the protection logic is disabled and not bypassed, it treats all transactions as failing the checks and enters the transactions into the Fault state. How the Firewall Component handles the transaction in the Fault state, depends on the values of the PE_ST.{FLT_CFG,ERR,RAZ} fields.

When the protection logic is enabled, the protection logic matches a transaction against the enabled regions of the Firewall Component. The Firewall Component checks the address and properties of the transaction, against the enabled regions of the Firewall Component. A region is enabled only when RGN_ST.EN bit is 1.

Note

Before performing the region matching, the Firewall Component checks that the transaction is wholly within the Protection Size of the Firewall Component. For more information on Protection Size, see section *C.4.1 Protection Size and bus address widths* on page Appx-C-305.

A transaction only matches if all the locations which it accesses, are contained within the region. It is permissible for a transaction to match in multiple regions at this stage. When performing the matching the Firewall Component uses the following:

- Transaction lower address (TLA) and upper address (TUA): Depending on the bus protocol, the protection logic needs to calculate the transaction lower and upper address from other properties of the bus protocol. For example, in AXI the lower and upper address is calculated by using the AxADDR, AxSIZE, AxLEN and AxBURST transaction properties.
- Region base address (RBA) and upper address (RUA): Depending on the configuration of the region, the protection logic needs to calculate the RUA using the RBA, size of the region, and whether it is configured as a power of 2 region.
- Address mask (MSK): This is calculated as follows:
  - When MULnPO2 is 0, the value is 0xFFFF_FFFF_FFFF_FFFF left shifted by the size of the region. If the size is set to `0x00` then the region is not checked.
  - When MULnPO2 is 1 the value is 0xFFFF_FFFF_FFFF_FFFF left shifted by MNRS.

Note

In both cases, the left shift causes 0s to be added in the least significant bit position.

The protection logic uses this expression when performing the region match:

Region Match = ((RBA & MSK) <= (TLA & MSK)) && ((TUA & MSK) <= (RUA & MSK))

Note

The Firewall always zero extends address to 64-bits.

After matching the transaction, with at least one region, the enabled MPEs of the regions which match are checked for an entry which has:

- The same MasterID as the master which issued the transaction
- The MPE has been programmed to match any MasterID, only for MPE0 of each region

Note

If the Firewall Component FC_CFG2.SINGLE_MST is 1 then the MasterID is ignored.

If a single MPE is found the transaction is checked against the MPL. Only enabled MPEs, of the regions which matched, are checked. An MPE is enabled when the associated RGN_ST.MPE_EN bit is 1. If there is more than one MPE found to match the transaction, a Programming Error is generated to software.

Note

If an MPE is programmed to match all MasterIDs, this is treated as if there was an MPE entry for each MasterID. This means that if an MPE is programmed to match all MasterIDs, and there is another MPE with the same MasterID as the transaction, a Programming Error is generated to software.

Software can program limits for the transactions allowed for a master using a combination of any of the supported permissions. For example, a region can read, write or execute permissions to Secure privileged, but only give Secure unprivileged read access and no access to both Non-secure privileged and unprivileged respectively.

The following figure shows an example, where a Firewall Component has 4 regions defined, regions 1 to 4.

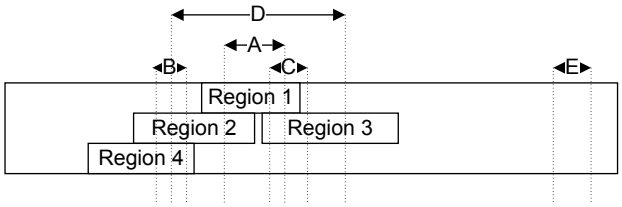

Figure C-5 Example of transaction matching a region

Transactions A to E are issued to the Firewall Component. The arrows indicate the range of address space which the transaction is attempting to access. The results of the region matching are as follows:

- Transaction A matches with region 1 and will use the MPEs associated with region 1 to perform the MPL check.
- Transaction B matches with regions 2 and 4. It will use both regions for the next step. However, only one of the regions can define an MPE with the MasterID associated with the transaction. If both regions define an MPE entry with the same MasterID, that results in a Programming Error being generated and transaction being treated as if it failed the checks.
- Transaction C matches with only region 3. It does not match with region 1 as the upper address of the transaction is greater than the upper address of the region.
- Transaction D does not match with any region, even though the address range it accesses is covered by region 1 to 4. This is because its lower address is below the base address for regions 1 and 3 and its upper address is above the upper address for regions 2 and 4.
- Transaction E matches no regions.

Note

The Firewall Component has an additional region - region 0, which is not shown in this diagram. The behavior of region 0 varies depending on the level of the Protection Extension implemented by the Firewall Component. See section *Default regions* on page Appx-C-311 for more information.

Note

Even though it is possible that two regions are defined to create a single contiguous region, a transaction which accesses across the boundary would not be contained exclusively within a region. In this case the transaction will not match either of the regions. Arm recommends that is avoided by either:

- Setting MNRS of a Firewall Component to be greater than or equal than the largest size transaction which a master can issue
- Software configuring the regions within the Firewall and the master, so that no transaction matches in two regions at once

C.4.3 Fault entries

This section describes fault entries.

A Firewall Component must have at least 1 fault entry.

Each fault entry provides details about the transactions which have entered the Faulted state. A fault entry can be:

- Valid: contains information about a faulting transaction which has not yet been acknowledged by software
- Invalid: contains UNKNOWN value

All fault entries start in the invalid state then transition into valid state when a faulting transaction's details are loaded. Once software acknowledges the fault entry, it transitions into invalid state. After this, the fault entry transitions between valid and invalid states on transaction fault and software acknowledgement respectively.

Fault entries are not accessed directly by software but are instead accessed using the FWE. Only valid fault entries are accessible in the FWE.

For information about how software can use the fault entries see *C.12.1 Fault usage model* on page Appx-C-394.

Fault types

A fault entry can contain either a:

- Transaction fault: details about a transaction which has failed the checks.
- Program fault: details about a transaction which has matched more than one region and MPE pairing.

In both cases the Firewall Component, if configured by the PE_ST.FLT_CFG, generates a fault entry. The only difference between the two fault types is the interrupt which is generated alongside the fault entry. For a transaction fault an Access Error interrupt is generated. For a programming fault a Programming Error interrupt is generated. Otherwise the same rules for fault entry properties, generation, acknowledgement and overflow, as described in *Fault entry properties* on page Appx-C-314 to *Fault entry overflow* on page Appx-C-315 apply.

Fault entry properties

A fault entry provides the following information:

- Address of the transaction
- Transaction properties:
  - MasterID
  - Privilege level, if FC_CFG1.PRIV_SPT is 1
  - Data or instruction, if FC_CFG1.INST_SPT is 1
  - Read or write
  - Security, if SE.1 is implemented
- Fault type

Fault entry generation

Depending on the value of PE_ST.FLT_CFG at the time the transaction is checked, an implementation must generate a fault entry.

See sections *PE_ST.FLT_CFG is 0b10* on page Appx-C-315 to *PE_ST.FLT_CFG is 0b11* on page Appx-C-315.

Depending on the bus protocol used for the Bus Slave interface and the implementation of the Firewall Component, it may be possible for a Firewall Component to process more than one transaction in a cycle. This can occur when:

- Read and write transactions are issued using different signals
- Within a single Bus Slave interface, transactions are allocated to a queue. Whereas transactions within different queues have no ordering requirements. Meanwhile transactions within the same queue are required to be processed in the order received.

When the protection logic processes more than one transaction in the same cycle, the order in which the fault entries are generated, for the parallel processed transactions, is IMPLEMENTATION DEFINED. It is also IMPLEMENTATION DEFINED, whether more than one fault entry per queue is generated at the same time. The protection logic must attempt to generate the fault entry for subsequent transactions in the queue when a transaction reaches the front of the queue.

Note

For two or more transactions, to be processed, at the same time there cannot be any dependency between the transactions.

PE_ST.FLT_CFG is 0b10

When a transaction faults and the PE_ST.FLT_CFG is `0b10`, the transaction enters the Faulted state.

The transaction is completed by the Firewall Component. Software has requested that a fault entry be generated. If when the protection logic attempts to generate the fault entry:

- All the fault entries are currently valid the protection logic does the following:
  - Transaction enters the Faulted state without generating a fault entry
  - Generates a Fault Entry Overflow interrupt
- At least one fault entry is invalid the protection logic does the following:
  - Transaction enters the Faulted state
  - Generates the fault entry. The fault entry must be added to the FWE before the terminated response is provided to the issuing master.
  - Generates an Access or Programming Error interrupt

Arm recommends that the Firewall Components attempt to generate a fault entry and Access or Programming Error interrupt, for transactions which enter the Faulted state, as soon as possible after the checks have been performed. This removes the need to hold the transaction information after the transaction has entered the Retired state.

PE_ST.FLT_CFG is 0b11

When a transaction faults and the PE_ST.FLT_CFG is `0b11` the transaction enters the Faulted state.

The transaction is completed by the Firewall Component. No fault entry or Access or Programming Error interrupt are generated.

Fault entry acknowledgement

A fault entry remains valid until software acknowledges the fault entry.

Fault entry overflow

A Firewall Component can generate more faults than it has fault entries. This leads to a Fault Entry Overflow and a Fault Entry Overflow interrupt is generated.

Fault entry and power

Fault entries are lost when the Firewall Component enters the Disconnected state.

Software can use the PE_CTRL.FE_PWR to set the PE_ST.FE_PWR field and prevent Firewall Components from entering the Disconnected state when the FWE contains a valid fault entry.

Fault window entry behavior

The FWE behaves as a FIFO with fault entries automatically added as they are generated, provided an invalid fault entry exists.

When software acknowledges a fault entry, it is removed from the FIFO and the FWE then points to the next valid fault entry, if one exists. If no more valid fault entries exist, the FWE:

- Indicates that there are no more valid fault entries, by the FE_CTRL.FE_VLD bit being 0
- FE_TAL, FE_TA and FE_TP all read as `0x0`
- Writes to the FE_CTRL.ACK field are ignored

The FE_CTRL register implements a field, FE_CTRL.LAST_FE, which indicates if the current fault entry is the last one in the FWE. When the FWE points to the last entry this field reads as 1, otherwise it reads as 0.

C.4.4 Transaction processing

The figure below shows the sequence of operations to process transactions as they arrive on the Bus Slave interface, of a Firewall Component which supports PE.1 or greater.

The default behavior is that a transaction enters the Faulted state unless it passes all the checks.

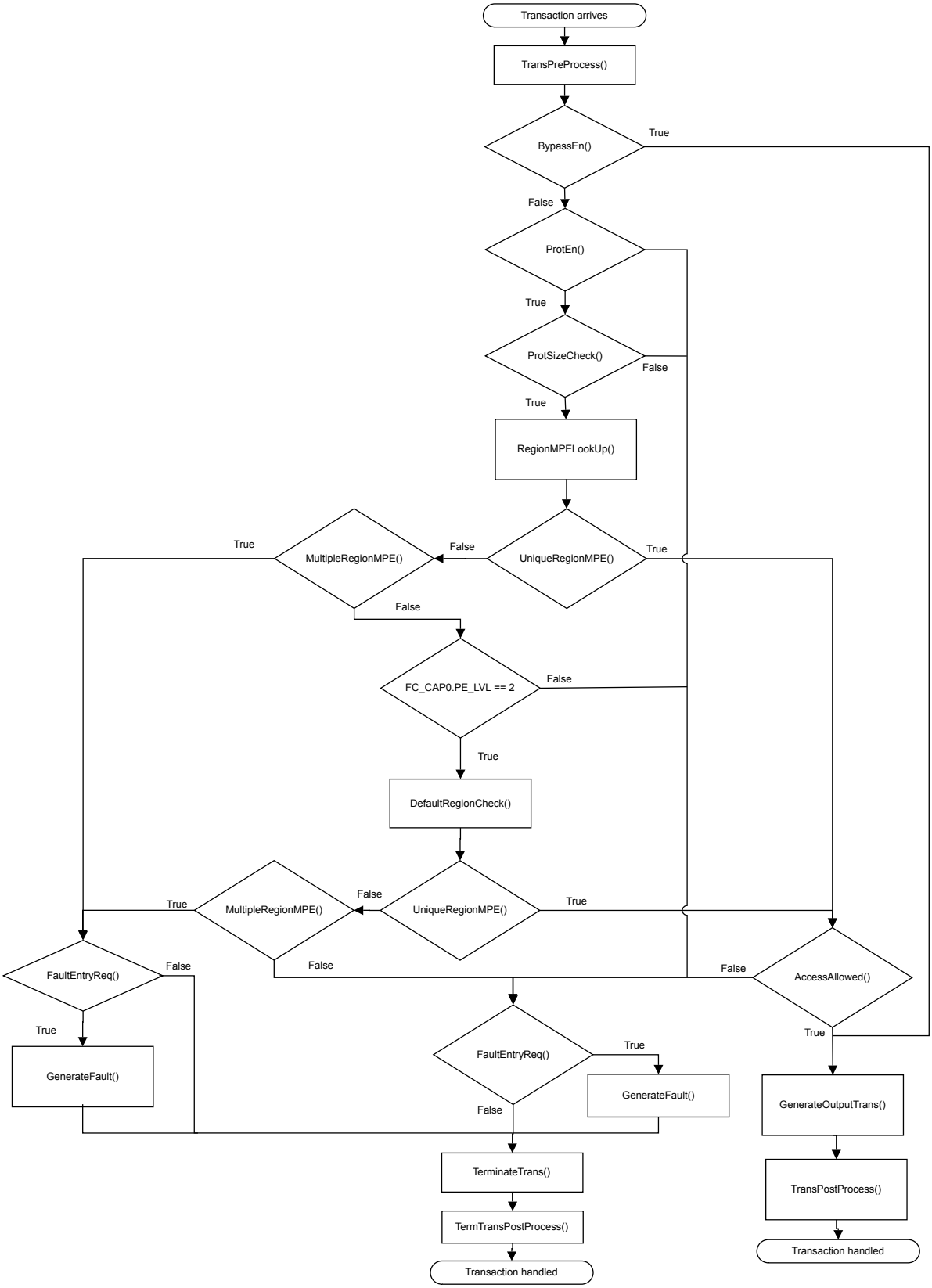

Figure C-6 Protection logic transaction processing flow

The following table describes the functions in the above figure.

Table C-20 Transaction processing functions

Processing function	Description
TransPreProcess()	Convert the incoming bus protocol into the format used by the protection logic. This function can block waiting for a new transaction.
TransPostProcess()	Convert the transaction format used by the protection logic, into the format used by the Bus Master interface.
TermTransPostProcess()	Convert the transaction format used by the Firewall Component, into the format used by the Bus Slave interface.
ProtEn()	This function checks whether the protection logic is enabled or not. It returns true if the logic is enabled otherwise it returns false.
BypassEn()	This function checks whether the Firewall Component protection logic is bypass or not. It returns true if the protection logic is bypassed otherwise it returns false.
ProtSizeCheck()	This function checks that the transaction is within the Protection Size defined for the Firewall Component's protection logic. It is applied when the transaction arrives, before any region lookup. It returns true when the transaction address is within the Protection Size, otherwise it returns false.
RegionMPELookUp()	This function performs the following operations: • Checks the transaction against all enabled regions, except for the default regions for Firewall Components which implemented PE.2, looking for a region which the transaction matches. • Checks the transaction against the MPEs, of the regions the transaction matches against, which have a MasterID the same as the transaction, or match any master. This function returns a count of the number of region and MPE pairs it matches against. When the transaction matches against a unique region MPE pair, the function returns the region and MPE it matches.
UniqueRegionMPE()	This function checks the results of the RegionMPELookUp() function. It returns true if a unique region and MPE pair was found. Otherwise it returns false.
MultipleRegionMPE()	This function checks the results of the RegionMPELookUp() function. It returns true if multiple region and MPE pairs are found. Otherwise it returns false.
DefaultRegionCheck()	The function is only implemented for Firewall Components, which implement PE.2. It performs the same operation as RegionMPELookUp() but only on the default region.
AccessAllowed()	This function takes the MPE found by either the RegionMPELookUp() or the DefaultRegionCheck() function and applies the permission check on the transaction. It returns true if the transaction is allowed, otherwise it returns false.
GenerateOutputTrans()	This function takes the incoming transaction and the translation information, from the region associated with the MPE used for the AccessAllowed() function, to generate the output transaction.
FaultEntryReq()	This function returns true if the value of PE_ST.FLT_CFG indicates a fault entry is required.
GenerateFault()	This function attempts to generate the fault entry.
TerminateTrans()	This function generates the correct response to a transaction which the Firewall Component has terminated.

C.4.5 Protection size interface

When a Firewall Component implements PE.2, a Protection Size interface is implemented on the Firewall Controller, allowing for the modification of the Protection Size of the Firewall Component.

The value of the Protection Size interface configures the Protection Size of the Firewall Component's protection logic, see section *C.4.1 Protection Size and bus address widths* on page Appx-C-305.

C.4.6 Bypass interface

The Firewall Component has a Bypass interface.

The value of the Bypass interface and the PE_ST.BYPASS_MSK field control whether the Firewall Component's protection logic is bypassed or not. Software can determine whether the Firewall Component's protection logic is bypassed or not using the PE_BPS.BYPASS_ST field. The table below shows how the values of the Bypass interface and PE_ST.BYPASS_MSK affect the value of PE_BPS.BYPASS_ST.

Table C-21 Firewall Component's protection logic behavior for bypass

Bypass interface	PE_ST.BYPASS_MSK	PE_BPS.BYPASS_ST
0b0	0b0	0b0
0b1	0b0	0b1
X	0b1	0b0

The value of PE_BPS.BYPASS_ST affects the behavior of all Firewall Components in the Firewall, as follows:

- 0b0: The Firewall Component is not bypassed. Transactions are treated as passing or failing the protection logic checks depending on the results of the region matching.
- 0b1: The Firewall Component is bypassed. Transactions are treated as passing the protection logic checks without performing a region matching. The transaction is then issued on a Bus Master interface of the Firewall Component, without any translation, if supported being applied.

C.4.7 Registers

This section summarizes the registers which are defined by the Protection Extension.

The Protection Extension defines three types of registers:

- Protection Control and Status
- Region Window Entry (RWE)
- Fault Window Entry (FWE)

The Firewall Component's protection logic can be either enabled or disabled by software. When the protection logic is enabled it processes transactions.

Alongside enabling and disabled the protection logic of the Firewall Component, the behavior of the Firewall Component when a transaction fails the checks can also be configured.

The Region Window Entry (RWE) is the interface which software uses to program the regions within the Firewall Component. The RWE occupies 21 consecutive 32-bit words. Software uses the RWE_CTRL register to select the region, within the Firewall Component the RWE refers to. For more information on the RWE_CTRL register, see section *RWE Control (RWE_CTRL)* on page Appx-C-324.

Software accesses the fault entries using the Fault Window Entry (FWE). The FWE occupies 5 consecutive 32-bit words. The FE_CTRL register allows software to acknowledge the fault entry referenced by the FWE.

Table C-22 Summary of Protection Extension registers

Offset	Short Name	Access	Name
Control and status			
0x100	*PE_CTRL* on page Appx-C-321	RW	Protection Extension Control
0x104	*PE_ST* on page Appx-C-322	RO	Protection Extension Status
0x108	*PE_BS* on page Appx-C-323	RW	Protection Extension Bypass

Table C-22 Summary of Protection Extension registers (continued)

Offset	Short Name	Access	Name
Region Window Entry (RWE)			
0x10C	*RWE_CTRL* on page Appx-C-324	RW	Region Window Entry Control
0x110	*RGN_CTRL0* on page Appx-C-324	RW	Region Control 0
0x114	*RGN_CTRL1* on page Appx-C-324	RW	Region Control 1
0x118	*RGN_LCTRL* on page Appx-C-325	RW	Region Lock Control
0x11C	*RGN_ST* on page Appx-C-326	RO	Region Status
0x120	*RGN_CFG0* on page Appx-C-328	RW	Region Config 0
0x124	*RGN_CFG1* on page Appx-C-328	RW	Region Config 1
0x128	*RGN_SIZE* on page Appx-C-329	RW	Region Size
0x12C	-	RO	Reserved
0x130	*RGN_TCFG0* on page Appx-C-330	RW	Region Translation Config 0
0x134	*RGN_TCFG1* on page Appx-C-330	RW	Region Translation Config 1
0x138	*RGN_TCFG2* on page Appx-C-331	RW	Region Translation Config 2
0x13C	-	RO	Reserved
0x140	*RGN_MID0* on page Appx-C-332	RW	Region MasterID 0
0x144	*RGN_MPL0* on page Appx-C-333	RW	Region Master Permission List 0
0x148	*RGN_MID1* on page Appx-C-332	RW	Region MasterID 1
0x14C	*RGN_MPL1* on page Appx-C-333	RW	Region Master Permission List 1
0x150	*RGN_MID0* on page Appx-C-332	RW	Region MasterID 2
0x154	*RGN_MPL2* on page Appx-C-333	RW	Region Master Permission List 2
0x158	*RGN_MID0* on page Appx-C-332	RW	Region MasterID 3
0x15C	*RGN_MPL3* on page Appx-C-333	RW	Region Master Permission List 3
Fault Window Entry (FWE)			
0x180	*FE_TAL* on page Appx-C-335	RO	Fault Entry Transaction Address Lower
0x184	*FE_TAU* on page Appx-C-335	RO	Fault Entry Transaction Address Upper
0x188	*FE_TP* on page Appx-C-335	RO	Fault Entry Transaction Properties
0x18C	*FE_MID* on page Appx-C-336	RO	Fault Entry MasterID
0x190	*FE_CTRL* on page Appx-C-336	RW	Fault Entry Control

Note

Some of the registers of the RWE are only implemented depending on the configuration and extensions which are implemented. For example, the RGN_TCFG{0-2} registers are only implemented when:

- RGN_TCFG{0-1} are implemented when RSE.1 or TE.2 are implemented.
- RGN_TCFG2 is implemented when TE.1 or greater is implemented.

When a register is not implemented in the RWE it is Reserved and generates a Configuration Access Error when accessed.

Protection Extension Control (PE_CTRL)

The following table gives a bit-level description of the Protection Extension Control (PE_CTRL) register.

The Protection Extension Control register allows software to enable or disable the Firewall Component's protection logic and select the behavior of the protection logic when a fault occurs.

Table C-23 PE_CTRL register

Bits	Name	Description	Type	Reset
[31]	EN	Request the Firewall Component's protection logic enables or disables. `0b0`: Request the Firewall Component's protection logic becomes disabled. `0b1`: Request the Firewall Component's protection logic becomes enabled. The reset value of this field is `0b0` for all Firewall Component's other than the Firewall Controller which resets to `0b1`. For more information on the Firewall Controller see section *C.11 Firewall Controller* on page Appx-C-373.	RW	See description
[30:6]	-	Reserved	RO	`0x000_0000`
[5]	BYPASS_MSK	Request Firewall Component behavior to Bypass interface. `0b0`: Firewall Component uses the value of the Bypass interface to calculate whether it is bypassed or not. `0b1`: Firewall Component ignores the value of the Bypass interface and treats the value as `0b0` to calculate whether it is bypassed or not.	RW	`0b0`
[4]	FE_PWR	Request Fault Entry power behavior `0b0`: Fault Entry does not prevent entry into a Disconnected state. `0b1`: Fault Entry does prevent entry into a Disconnected state.	RW	`0b0`
[3:2]	FLT_CFG	Requested Fault Configuration Configures the behavior of the Firewall Component when a transaction enters the Faulted state. `0b00`: Reserved and treated as 10 `0b01`: Reserved and treated as 10 `0b10`: Terminate transaction, generate a fault entry and Access or Programming Error interrupt. `0b11`: Terminate transaction, but no fault entry or Access or Programming Error interrupt are generated.	RW	`0b10`
[1]	RAZ	Requested behavior for read data returned for read transactions terminated by the Firewall Component. `0b0`: Read data is based on the StreamID `0b1`: Read data all 0s	RW	`0b0`
[0]	ERR	Requested behavior for responses for transactions terminated by the Firewall Component. `0b0`: No error `0b1`: Error	RW	`0b1`

Arm recommends the following:

- No outstanding transactions are being processed by the Firewall Component when the values of the PE_CTRL.{FLT_CFG,RAZ,ERR} are changing (for Firewall Components other than the Firewall Controller). The method which software uses to guarantee this, is outside the scope of Appendix C .
- There are no outstanding transactions, except for configuration accesses to change the value of the PE_CTRL.{FLT_CFG,RAZ,ERR}, when Firewall Controller fields are changed.

Firewall Component Status (PE_ST)

The following table gives a bit-level description of the Firewall Component Status (PE_ST) register.

Table C-24 PE_ST register

Bits	Name	Description	Type	Reset
[31]	EN	Status of the Firewall Component's protection logic. `0b0`: Firewall Component's protection logic is disabled. `0b1`: Firewall Component's protection logic is enabled. When SRE.1 is implemented and the Firewall Component has entered the Disconnected state this field matches the value in PE_CTRL.EN. As if the request to change from enabled to disabled or disabled to enabled has completed. The reset value matches the reset value of PE_CTRL.EN.	RO	See description
[30:6]	-	Reserved	RO	`0x0000_000`
[5]	BYPASS_MSK	Firewall Component behavior to Bypass interface. `0b0`: Firewall Component uses the value of the Bypass interface to calculate whether it is bypassed or not. `0b1`: Firewall Component ignores the value of the Bypass interface and treats the value as `0b0` to calculate whether it is bypassed or not.	RO	0
[4]	FE_PWR	Fault Entry power behavior. `0b0`: Fault Entry does not prevent entry into a Disconnected state. `0b1`: Fault Entry does prevent entry into a Disconnected state.	RO	`0b0`
[3:2]	FLT_CFG	Fault Configuration Behavior of the Firewall Component when a transaction enters the Faulted state. `0b00`: Reserved `0b01`: Reserved `0b10`: Terminate transaction, generate a fault entry and Access or Programming Error interrupt. `0b11`: Terminate transaction, but no fault entry or Access or Programming Error interrupt are generated.	RO	`0b10`

Table C-24 PE_ST register (continued)

Bits	Name	Description	Type	Reset
[1]	RAZ	Value returned for read accesses when the Firewall Component terminates the transaction. `0b0`: Read data is based on the StreamID `0b1`: Read data is all 0s	RO	`0b0`
[0]	ERR	Response for a transaction terminated by the Firewall Component. `0b0`: No error `0b1`: Error	RO	`0b1`

The value of PE_ST.EN defines whether the protection logic is enabled or disabled. When the protection logic is disabled, transactions do not match against the regions of the Firewall Component and all enter the Fault state.

The values of PE_ST.{FLT_CFG,RAZ,ERR} define the fault behavior of the Firewall Component's protection logic. The reset values of the PE_ST.{FLT_CFG,RAZ,ERR} fields match the reset values of the associated field in the PE_CTRL register.

Note

See section *C.4.9 Protection logic terminated transaction response* on page Appx-C-341 for more information on the transaction response generated for transactions terminated by the Firewall Component's protection logic.

Protection Extension Bypass (PE_BPS)

The following table gives a bit-level description of the Protection Extension Bypass (PE_BPS) register.

Table C-25 PE_BPS register

Bits	Name	Description	Type	Reset
[31]	BYPASS_VLD	Indicates whether the values in the BYPASS_ST and BYPASS_IF_ST is valid or not. `0b0`: Values are not valid `0b1`: Values are valid The behavior of this field depends the level of SRE implemented by the Firewall. SRE.0: This field always reads as `0b1` SRE.1: This field reads as `0b1` only when the Firewall Component is in the Connected state. Otherwise this field reads as `0b0`.	RO	See description
[30:2]	-	Reserved	RO	`0x0000_0000`

Table C-25 PE_BPS register (continued)

Bits	Name	Description	Type	Reset
[1]	BYPASS_ST	Indicates if the Firewall Component's protection logic is bypassed or not. `0b0`: Firewall Component's protection logic is not bypassed. `0b1`: Firewall Component's protection logic is bypassed.	RO	0
[0]	BYPASS_IF_ST	Bypass interface status The reset value of this field depends on the value of the Bypass interface of the Firewall Component.	RO	See description

RWE Control (RWE_CTRL)

The following table gives a bit-level description of the RWE Control (RWE_CTRL) register.

Table C-26 RWE Control register description

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	`0x00_0000`
[7:0]	RGN_INDX	Region Index Selects the region to which the RWE refers to. The width of this field is dependent on the log2(FC_CFG1.NUM_RGN+1) rounded up to the nearest whole number. Any unused bits are Reserved and treated as RAZ/WI.	RW	`0x00`

Configuration Access Errors are generated for any access to any register of the RWE, when the RWE_CTRL.RGN_INDEX refers to a region which is not implemented by the Firewall Component. For example, a Firewall Component with 6 regions has an RGN_INDX field width of 3. This allows software to program the values 0 to 7, however only regions 0 to 5 exists within the Firewall Component.

Region Control 0 (RGN_CTRL0)

The following table gives a bit-level description of the Region Control 0 (RGN_CTRL0) register.

Table C-27 Region Control 0 register

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved	RO	`0x0000_0000`
[0]	EN	Region enable `0b0`: Request to disable the region `0b1`: Request to enable the region	RW	0

Region Control 1 (RGN_CTRL1)

The following table gives a bit-level description of Region Control 1 (RGN_CTRL1) register.

Table C-28 RGN_CTRL1 register

Bits	Name	Description	Type	Reset
[31:5]	-	Reserved	RO	0x000_0000
[4]	MPE3_EN	Master enable for MPE3 0b0: Request to disable Master permission entry 3. 0b1: Request to enable Master permission entry 3. Only implemented when FC_CFG2.NUM_MPE is 3. Otherwise this field is Reserved and treated as RAZ/WI.	RW	0
[3]	MPE2_EN	Master enable for MPE2 0b0: Request to disable Master permission entry 2. 0b0: Request to enable Master permission entry 2. Only implemented when FC_CFG2.NUM_MPE is 2 or greater. Otherwise this field is Reserved and treated as RAZ/WI.	RW	0
[2]	MPE1_EN	Master enable for MPE1 0b0: Request to disable Master permission entry 1. 0b1: Request to enable Master permission entry 1. Only implemented when FC_CFG2.NUM_MPE is 1 or greater. Otherwise this field is Reserved and treated as RAZ/WI.	RW	0
[1]	MPE0_EN	Master enable for MPE0 0b0: Request to disable Master permission entry. 0 0b1: Request to enable Master permission entry 0.	RW	0
[0]	-	Reserved	RO	0

Region Lock Control (RGN_LCTRL)

The following table gives a bit-level description of the Region Lock Control (RGN_LCTRL) register.

Table C-29 RGN_LCTRL register

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved	RO	`0x0000_0000`
[0]	LOCK	Control the lock status of the region. `0b0`: Region is unlocked `0b1`: Region is locked When this field is 1 a configuration access which attempts to update any of the following fields: • RGN_CTRL{0-1} • RGN_CFG{0-1} • RGN_SIZE • RGN_TCFG{0-2} • RGN_MID{0-3} • RGN_MPL{0-3} Generates: • A Configuration Access Error • A Tamper interrupt and Tamper report, if there is no valid Tamper report present. Otherwise, a Tamper Overflow interrupt is generated. For more information see *C.11.3 Lockdown Extension* on page Appx-C-378. This field is Reserved and treated as RAZ/WI when LDE.0 or LDE.1 is implemented. When any of the following occur: • Firewall Components enter the Full lockdown state • Firewall Components enter the Partial lockdown state and this field is set to 1 This field becomes RO and any attempt to update this field generates: • A Configuration Access Error • A Tamper interrupt and Tamper report, if there is no valid Tamper report present. Otherwise, a Tamper Overflow interrupt is generated. For more information see section *C.11.3 Lockdown Extension* on page Appx-C-378. For more information on the Firewall Component lockdown state see *C.9.1 Firewall Component lockdown* on page Appx-C-366.	RW	`0x0`

Region Status (RGN_ST)

The following table gives a bit-level description of the Region Status (RGN_ST) register.

Table C-30 RGN_ST register

Bits	Name	Description	Type	Reset
[31:5]	-	Reserved	RO	0x000_0000
[4]	MPE3_EN	Master enable for MPE3 0b0: Master permission entry 3 is disabled 0b1: Master permission entry 3 is enabled Only when a master permission entry is enabled will an incoming transaction be allowed to match against the entry. When this field is 1, RGN_MID3 and RGN_MPL3 are read-only and attempts to update the registers will generate a Configuration Access Error.	RO	0
[3]	MPE2_EN	Master enable for MPE2 0b0: Master permission entry 2 is disabled 0b1: Master permission entry 2 is enabled Only when a master permission entry is enabled will an incoming transaction be allowed to match against the entry. When this field is 1, RGN_MID2 and RGN_MPL2 are read-only and attempts to update the registers will generate a Configuration Access Error.	RO	0
[2]	MPE1_EN	Master enable for MPE1 0b0: Master permission entry 1 is disabled. 0b1: Master permission entry 1 is enabled. Only when a master permission entry is enabled will an incoming transaction be allowed to match against the entry. When this field is 1, RGN_MID1 and RGN_MPL1 are read-only and attempts to update the registers will generate a Configuration Access Error.	RO	0

Table C-30 RGN_ST register (continued)

Bits	Name	Description	Type	Reset
[1]	MPE0_EN	Master enable for MPE0: `0b0`: Master permission entry 0 is disabled. `0b1`: Master permission entry 0 is enabled. Only when a master permission entry is enabled will an incoming transaction be allowed to match against the entry. When this field is 1, RGN_MID0 and RGN_MPL0 are read-only and attempts to update the registers will generate a Configuration Access Error.	RO	0
[0]	EN	Region enable: `0b0`: Region disable `0b1`: Region enable When set to 1 the following registers are read-only: • RGN_CFG0 • RGN_CFG1 • RGN_TCFG0 • RGN_TCFG1 • RGN_TCFG2 • RGN_SIZE Any attempt to write to these registers, when the region is enabled, generates a Configuration Access Error.	RO	0

Region Config {0,1} (RGN_CFG{0,1})

The following table gives a bit-level description of the Region Config {0,1} register.

The base address of a region is programmed using the RGN_CFG{0,1} registers, with the lower 32-bits of the address being in RGN_CFG0 and the upper 32-bits in RGN_CFG1.

Table C-31 RGN_CFG{0,1} register

Bits	Name	Description	Type	Reset
[63:5]	BASE_ADDR	The base address of the region. The width of this field depends on the MXRS and MNRS properties for the Firewall Component. Number of bits implemented is log2(MXRS)-1 to log2(MNRS), starting at bit log2(MNRS). Any unimplemented bits are Reserved and treated as RAZ/WI. If MXRS and MNRS are equal, then all bits in this register are Reserved and treated as RAZ/WI.	See below	See below
[4:0]	-	Reserved	RO	`0x00`

Depending on the Firewall Component type, the access type and reset value of BASE_ADDR is:
- Read-write with an UNKNOWN reset value for a Firewall Component, implementing PE.2
- Read-only with a reset value equal to the base address of the region, bits log2(MXRS)-1 to log2(MNRS) only, for Firewall Component, implementing PE.1

Note

If either RGN_CFG{0,1} registers have all bits Reserved and treated as RAZ/WI, then the register is Reserved.

Region Size (RGN_SIZE)

The following table gives a bit-level description of the Region Size (RGN_SIZE) register.

Table C-32 RGN_SIZE register

Bits	Name	Description	Type	Reset
[31:9]	-	Reserved	RO	`0x00_0000`
[8]	MULnPO2	Selects whether the region is defined with a base address and size or a base and upper address. `0b0`: Region defined by a base address and size, which must be a power of 2. The value of RGN_SIZE.SIZE is used. `0b1`: Region defined by a base and upper address, which must both be an integer multiple of MNRS. The value of RGN_SIZE.SIZE is ignored. This field is Reserved and treated as RAZ/WI when RSE.0 is implemented.	See below	See below
[7:0]	SIZE	The size of the region. `0x00`: 0B `0x05`: 32B `0x06`: 64B … `0x0C`: 4KB `0x0D`: 8KB … `0x40`: 16KB The legal values this field can be set to depend on the MNRS and MXRS of the Firewall Component. If software attempts to set this field to a value which is: • Greater than `0x40` • Less than MNRS of the Firewall. Component • Greater than MXRS of the Firewall Component The field is set to `0x00` instead. When this field reads as `0x00` does not match against any transactions. The value of the RGN_SIZE field is only used if RGN_SIZE.MULnPO2 is 0. Otherwise the field is ignored.	See below	See below

Depending on the level of Protection Extension implemented:

- MULnPO2 field is:
  - When PE.2 is implemented:

    - Read-write with an UNKNOWN reset value for all regions, except for the default region
    - Read-only with a reset value of 0 for the default region
  - When PE.1 is implemented:
    - Read-only with a reset value set at design time

- SIZE field is:
  - When PE.2 is implemented:
    - Read-write with an UNKNOWN reset value for all regions, except for the default region
    - Read-only with a reset value which matches the value in the FC_CFG2.PROT_SIZE field, for the default region
  - When PE.1 is implemented and MULnPO2 is set to 0:
    - Read-only with a reset value set at design time
  - When PE.1 is implemented and MULnPO2 is set to 1:
    - Read-only with an UNKNOWN reset value

Region Translation Config {0,1} (RGN_TCFG{0,1})

The following table gives a bit-level description of the Region Translation Config {0,1} register.

The output address or the upper address of a region is programmed using the RGN_TCFG{0,1} registers, with the lower 32-bits of the address being in RGN_TCFG0, and the upper 32-bits in RGN_TCFG1.

These registers are implemented when either RSE.1 or TE.2 is implemented, otherwise they are Reserved and generate a Configuration Access Error when accessed.

Table C-33 RGN_TCFG{0,1} register

Bits	Name	Description	Type	Reset
[63:5]	OUTPUT_ADDR/ UPPER_ADDR	The output address or the upper address range of the region. The width of this field depends on the MXRS and MNRS properties for the Firewall Component. Number of bits implemented is log2(MXRS)-1 to log2(MNRS), starting at bit log2(MNRS). Any unimplemented bits are Reserved and treated as RAZ/WI.	See below	See below
[4:0]	-	Reserved	RO	`0x00`

Depending on the Firewall Component configuration, the OUTPUT_ADDR/UPPER_ADDR field behaves as follows:

- When PE.2 is implemented:
  - Read-write field with an UNKNOWN reset value for all regions, except for the default region.
  - Read-only field with an UNKNOWN reset value for the default region.
- When PE.1 and TE.0 or TE.1 are implemented and RGN_SIZE.MULnPO2 is 0:
  - Read-only field with an UNKNOWN reset value.
- When PE.1 and TE.2 are implemented and RGN_SIZE.MULnPO2 is 0:
  - Read-write field with an UNKNOWN reset value.
- When PE.1 is implemented and RGN_SIZE.MULnPO2 is 1:
  - Read-only field with a reset value set at design time.

——— **Note** ———

If either RGN_TCFG{0,1} registers have all bits Reserved and treated as RAZ/WI, then the register is Reserved.

———————————

Region Translation Config 2 (RGN_TCFG2)

The following table gives a bit-level description of the Region Translation Config 2 (RGN_TCFG2) register.

This register is implemented when TE.1 or greater is implemented, otherwise it is Reserved and generates a Configuration Access Error when accessed.

Table C-34 RGN_TCFG2 register

Bits	Name	Description	Type	Reset
[31:18]	-	Reserved	RO	`0x0000`
[17]	ADDR_TRANS_EN	Address Translation enable `0b0`: Address translation is disabled. The output transaction has the same address as the incoming transaction. `0b1`: Address translation is enabled. The output transaction has an address is set based on the formula in *Output transaction translation address* on page Appx-C-356 This field is Reserved and treated as RAZ/WI when: • TE.1 or lower is implemented • PE.2 is implemented for the Default Region • Regions 0-2 for Firewall Controller	RW	`0b0`
[16]	MA_TRANS_EN	Memory Attribute Translation enable `0b0`: Memory attribute translation is disabled. The output transaction has the same memory attribute as the incoming transaction. `0b1`: Memory attribute translation is enabled. The output transaction's memory attribute is as defined in RGN_TCG2.MA. This field is Reserved and treated as RAZ/WI when either: • FC_CFG1.MA_SPT is `0b0` • TE.0 is implemented	RW	`0b0`
[15:14]	INST	Output transaction instruction or data: `0b00`: Use incoming transaction value. `0b01`: Reserved and treated as 00. `0b10`: Data `0b11`: Instruction This field only affects read transactions. All write transactions are considered as data and outgoing write transactions are outputted as data access. This field is Reserved and treated as RAZ/WI when either: • FC_CFG1.INST_SPT is `0b0` • TE.0 is implemented	RW	`0b00`

Table C-34 RGN_TCFG2 register (continued)

Bits	Name	Description	Type	Reset
[13:12]	PRIV	Output transaction privileged level `0b00`: Use incoming privileged level. `0b01`:Reserved and treated as 00. `0b10`: Unprivileged `0b11`: Privileged This field is Reserved and treated as RAZ/WI when either: • FC_CFG1.PRIV_SPT is `0b0` • TE.0 is implemented	RW	`0b00`
[11:4]	MA	Output transaction memory attribute Defines the memory type, cache allocation policy and whether it is transient or not for the output transaction. For the list of values see section *Output transaction memory attribute property* on page Appx-C-357. This field is Reserved and treated as RAZ/WI when either: • FC_CFG1.MA_SPT is `0b0` • TE.0 is implemented	RW	`0x00`
[3:2]	SH	Output transaction shareability. `0b00` – Non-shareable. `0b01` – Use incoming shareability. `0b10`: Outer shareable. `0b11`: Inner shareable. This field is Reserved and treated as WI and as reads as `0b01`, when either: FC_CFG1.SH_SPT is 0. or TE.0 is implemented.	RW	`0b01`
[1:0]	NS	Output transaction security `0b00` – Output transaction is marked with the same security as incoming transaction. `0b01`: Reserved and treated as `0b00`. `0b10`: Output transaction is marked as Secure. `0b11`: Output transaction is marked as Non-secure. This only applies if the incoming transaction was Secure. This field is Reserved and treated as RAZ/WI when any of the following are true: • FC_CFG1.SEC_SPT is `0b0` • TE.0 is implemented	RW	`0b00`

When the Firewall Component implements PE.1 and the region has MULnPO2 set to `0b1`, all fields in this register are read-only with a reset value as defined in the table above.

Region MasterID {0-3} (RGN_MID{0-3})

The following table gives a bit-level description of the Region MasterID {0-3} (RGN_MID{0-3}) register.

The number of Region MasterID implemented, per region, depends on the value of the FC_CFG1.NUM_MPE. When a Region MasterID is not implemented it is Reserved and generates a Configuration Access Error when accessed.

Table C-35 RGN_MID{0-3} register

Bits	Name	Description	Type	Reset
[31:0]	MST_ID	MasterID. The value of the MasterID part of the StreamID which the transaction must have to match this MPE. The width of this field depends on the value of FC_CFG2.MST_ID_WIDTH. Any unused bits are Reserved and treated as RAZ/WI. This field is read-only when FC_CFG2.SINGLE_MST is `0b1`.	RW	The reset value of this field depends on the value of FC_CFG2.SINGLE_MST: `0b0`: UNKNOWN `0b1`: MST_ID is a fixed value defined at design time

Region Master Permission List {0-3} (RGN_MPL{0-3})

The following table gives a bit-level description of the Region Master Permission List {0-3} (RGN_MPL{0-3}) register.

The number of Region Master Permission Lists implemented per region, depends on the value of the FC_CFG1.NUM_MPE. When a Region Master Permission List is not implemented, it is Reserved and generates a Configuration Access Error when accessed.

Table C-36 RGN_MPL{0-3} register

Bits	Name	Description	Type	Reset
[31:13]	-	Reserved	RO	`0x0_0000`
[12]	ANY_MST	Selects whether entry is used for all transactions, irrespective of MasterID. `0b0`: MPE only used if MasterID of transaction and MPE match `0b1`: MPE used irrespective of transaction MasterID This field is read-only when FC_CFG2.SINGLE_MST is 1. This field is Reserved and treated as RAZ/WI for RGN_MPL{1-3}.	RW	The reset value of this field depends on the value of FC_CFG2.SINGLE_MST: `0b0`: UNKNOWN `0b1`: ANY_MST is 0
[11]	SPX	Secure privilege execute enable `0b0`: Secure privileged instruction fetches are not allowed. `0b1`: Secure privileged instruction fetches are allowed. This field is Reserved and treated as RAZ/WI when any of the following are true: • FC_CFG1.SEC_SPT is 0 • FC_CFG1.PRIV_SPT is 0 • FC_CFG1.INST_SPT is 0	RW	UNKNOWN
[10]	SPW	Secure privilege write enable `0b0`: Secure privileged data write operations are not allowed. `0b1`: Secure privileged data write operations are allowed. This field is Reserved and treated as RAZ/WI when any of the following are true: • FC_CFG1.SEC_SPT is 0 • FC_CFG1.PRIV_SPT is 0	RW	UNKNOWN

Table C-36 RGN_MPL{0-3} register (continued)

Bits	Name	Description	Type	Reset
[9]	SPR	Secure privilege read enable `0b0`: Secure privileged data read operations are not allowed. `0b1`: Secure privileged data read operations are allowed. This field is Reserved and treated as RAZ/WI when any of the following are true: • FC_CFG1.SEC_SPT is 0 • FC_CFG1.PRIV_SPT is 0	RW	UNKNOWN
[8]	SUX	Secure unprivileged execute enable `0b0`: Secure unprivileged instruction fetches are not allowed. `0b1`: Secure unprivileged instruction fetches are allowed. This field is Reserved and treated as RAZ/WI when any of the following are true: • FC_CFG1.SEC_SPT is 0 • FC_CFG1.INST_SPT is 0	RW	UNKNOWN
[7]	SUW	Secure unprivileged write enable `0b0`: Secure unprivileged data write operations are not allowed. `0b1`: Secure unprivileged data write operations are allowed. This field is Reserved and treated as RAZ/WI when FC_CFG1.SEC_SPT is 0.	RW	UNKNOWN
[6]	SUR	Secure unprivileged read enable `0b0`: Secure unprivileged data read operations are not allowed. `0b1`: Secure unprivileged data write operations are allowed. This field is Reserved and treated as RAZ/WI when FC_CFG1.SEC_SPT is 0.	RW	UNKNOWN
[5]	NSPX	Non-secure privilege execute enable `0b0`: Non-secure privileged instruction fetches are not allowed. `0b1`: Non-secure privileged instruction fetches are allowed. This field is Reserved and treated as RAZ/WI when any of the following are true: • FC_CFG1.PRIV_SPT is 0 • FC_CFG1.INST_SPT is 0	RW	UNKNOWN
[4]	NSPW	Non-secure privilege write enable `0b0`: Non-secure privileged data write operations are not allowed. `0b1`: Non-secure privileged data write operations are allowed. This field is Reserved and treated as RAZ/WI when FC_CFG1.PRIV_SPT is 0.	RW	UNKNOWN

Table C-36 RGN_MPL{0-3} register (continued)

Bits	Name	Description	Type	Reset
[3]	NSPR	Non-secure privilege read enable `0b0`: Non-secure privileged data read operations are not allowed. `0b1`: Non-secure privileged data read operations are allowed. This field is Reserved and treated as RAZ/WI when FC_CFG1.PRIV_SPT is 0.	RW	UNKNOWN
[2]	NSUX	Non-secure unprivileged execute enable `0b0`: Non-secure unprivileged instruction fetches are not allowed. `0b1`: Non-secure unprivileged instruction fetches are allowed. This field is Reserved and treated as RAZ/WI when FC_CFG1.INST_SPT is 0.	RW	UNKNOWN
[1]	NSUW	Non-secure unprivileged write enable `0b0`: Non-secure unprivileged data write operations are not allowed. `0b1`: Non-secure unprivileged data write operations are allowed	RW	UNKNOWN
[0]	NSUR	Non-secure unprivileged read enable `0b0`: Non-secure unprivileged data read operations are not allowed. `0b1`: Non-secure unprivileged data read operations are allowed.	RW	UNKNOWN

Fault Entry Transaction Address Lower (FE_TAL)

The following table gives a bit-level description of the Fault Entry Transaction Address Lower (FE_TAL) register.

Table C-37 FE_TAL register

Bits	Name	Description	Type	Reset
[31:0]	FAULT_ADDR_LWR	Fault transaction address lower. This field is RAZ when FE_CTRL.FE_VLD is 0.	RO	UNKNOWN

Fault Entry Transaction Address Upper (FE_TAU)

The following table gives a bit-level description of the Fault Entry Transaction Address Upper register.

Table C-38 FE_TAU register

Bits	Name	Description	Type	Reset
[31:0]	FAULT_ADDR_UPR	Fault transaction address upper. This field is RAZ when FE_CTRL.FE_VLD is 0.	RO	UNKNOWN

Fault Entry Transaction Properties (FE_TP)

The following table gives a bit-level description of the Fault Entry Transaction Properties register.

Table C-39 FE_TP register

Bits	Name	Description	Type	Reset
[31:22]	-	Reserved	RO	0x0000
[21]	W	Indicates whether the transaction was a read or write: 0b0: Read 0b1: Write	RO	UNKNOWN
[20:19]	-	Reserved	RO	0x00
[18]	INST	Indicates whether the transaction was an instruction or data access: 0b0: Data 0b1: Instruction	RO	UNKNOWN
[17]	PRIV	Indicates the privileged level of the transaction: 0b0Unprivileged 0b1: Privileged	RO	UNKNOWN
[16]	NS	Indicates the security level of the transaction: 0b0: Secure 0b1: Non-secure When SE.0 is implemented this field is Reserved and is treated as RAO/WI.	RO	UNKNOWN
[15:0]	-	Reserved	RO	0x0000

When FE_CTRL.FE_VLD is 0, this register reads as zero.

Fault Entry MasterID (FE_MID)

The following table gives a bit-level description of the Fault Entry MasterID (FE_MID) register.

Table C-40 FE_MID register

Bits	Name	Description	Type	Reset
[31:0]	MST_ID	The reset value of this field depends on the value of FC_CFG2.SINGLE_MST: Indicates the MasterID of the master which issued the transaction. The width of this field depends on the value of FC_CFG2.MST_ID_WIDTH. Any unused bits are Reserved and treated as RAZ/WI.	RO	0b0: UNKNOWN 0b1: is a fixed value defined at design time.

When FE_CTRL.FE_VLD is 0b0 this register reads as zero.

Fault Entry Control (FE_CTRL)

The following table gives a bit-level description of the Fault Entry Control (FE_CTRL) register.

Table C-41 FE_CTRL register

Bits	Name	Description	Type	Reset
[31]	LAST_FE	Indicates if this fault entry is the last valid entry: `0b0`: Entry is the not last valid fault entry. `0b1`: Entry is the last valid fault entry.	RO	`0x0`
[30]	FE_VLD	Indicates whether the FWE is pointing to a valid fault entry: `0b0`: FWE is pointing to an invalid fault entry. `0b1`: FWE is pointing to a valid fault entry. When this field is `0b0` the following fields, in the FWE, read as `0x0`: • FE_TAL • FE_TAU • FE_TP • FE_MID • FE_CTRL.FLT_TYPE	RO	`0x0`
[29:4]	-	Reserved	RO	`0x000_0000`
[3]	FLT_TYPE	Indicates the fault type: `0b0`: Transaction fault `0b1`: Programming fault	RO	`0x0`
[2:1]	-	Reserved	RO	`0x00`
[0]	ACK	Acknowledge the transaction. This field always reads as `0b0`. Writes to this field behave as follows: `0b0`: Ignored `0b1`: Fault transaction is acknowledged Writes to this register are ignored if FE_CTRL.FE_VLD is `0b0`.	WO	`0x0`

When software acknowledges a fault entry, the FWE points to the next fault entry.

C.4.8 Changing Configuration Settings of Protection Logic

This section covers the requirements for changing the configuration settings of the protection logic and individual regions.

Enabling and disabling the Firewall Component's protection logic

The Firewall Component's protection logic can be enabled and disabled using the PE_CTRL.EN field. .

The status of the Firewall Component's protection logic is indicated by the PE_ST.EN field. When PE_CTRL.EN and PE_ST.EN have different values. It indicates that the Firewall Component is either performing an enablement or disablement of the protection logic:

- Enablement: PE_CTRL.EN is `0b1` and PE_ST.EN is `0b0`
- Disablement: PE_CTRL.EN is `0b0` and PE_ST.EN is `0b1`

The value of PE_ST.EN only updates to the value in the PE_CTRL.EN field, when the enablement or disablement process completes:

- The enablement process completes when the write to the PE_CTRL.EN completes.
- The disablement process completes when any new transaction detects the protection logic has been disabled and will be treated as failing the checks.

If software attempts to set the PE_CTRL.EN field back to its previous value while an enablement or disablement is in-progress, the Firewall treats the update as a Configuration Access Error and does not update the PE_CTRL register.

Arm strongly recommends:

- When software has set the value of PE_CTRL.EN, it waits for the value to be reflected in PE_ST.EN before changing the value again.
- That before enabling the protection logic, software waits for any previous configuration of the Firewall Component to complete. If software enables the Firewall Component protection logic while a previous configuration of the Firewall Component's protection logic is ongoing, it is UNPREDICTABLE whether processed transactions use the new or old configuration values.
- Software waits for the disablement process to complete before attempting any other configuration of the Firewall Component's protection logic. If software attempts to update the configuration of the Firewall Component's protection logic, before the disablement process completes, it is UNPREDICTABLE whether transactions which are processed, see the new or old configuration values.

Changing fault entry power behavior

The Firewall Component behavior can be configured for when the following conditions occur:

- Firewall Component has at least one valid fault entry
- The Power Control interface of the Firewall Component, makes a request to enter a power mode where the Firewall Component would be considered non-operational

Using the PE_CTRL.FE_PWR field, software can configure whether the Firewall Component accepts or denies the power request under these conditions. Upon writing this field, the PE_ST.FE_PWR field is updated automatically to match the value written to the PE_CTRL.FE_PWR field. The Firewall Component uses the value in the PE_ST.FE_PWR to select whether it accepts or denies the power request.

Changing fault transaction behavior

The fault behavior of the Firewall Component can be configured by software, using the PE_CTRL.FLT_CFG field.

The current fault behavior of the Firewall Component, is indicated in the PE_ST.FLT_CFG field.

When the value of the PE_CTRL.FLT_CFG is different to the PE_ST.FLT_CFG field, the Firewall Component is changing its fault behavior. If software attempts to change the value of the PE_CTRL.FLT_CFG field, whilst the Firewall Component is changing its fault behavior, Firewall treats the update as a Configuration Access Error and does not update the PE_CTRL register.

The value of the PE_ST.FLT_CFG field updates to the value in the PE_CTRL.FLT_CFG field, when the Firewall Component treats any new transaction faults as configured in the PE_CTRL.FLT_CFG field. Any transaction which has previously been faulted, continues to behave as defined by the value of PE_ST.FLT_CFG at the point when it failed the checks. It is UNPREDICTABLE whether a transaction which fails the protection logic checks, while the values of PE_CTRL.FLT_CFG and PE_ST.FLT_CFG are different, uses the new or old value of PE_ST.FLT_CFG.

Arm strongly recommends:

- When software changes the value of either PE_CTRL.FLT_CFG, it waits for the value of PE_ST.FLT_CFG to reflect this change, before changing the value again.
- When software changes the values of PE_CTRL.FLT_CFG there are no outstanding transactions to the Firewall Component. The software method used to achieve this is outside the scope of this

document. If there are outstanding transactions, it is UNPREDICTABLE whether faulted transactions use the new or old value of the PE_ST.FLT_CFG field.

Note

The PE_CTRL.FLT_CFG field has Reserved values which are treated as other values. The Firewall Component can update the value of PE_ST.FLT_CFG with no delay, if the Reserved value written is treated as the current value of the PE_ST.FLT_CFG. For example, `0b01` is treated as `0b10` if the current value of PE_ST.FLT_CFG is `0b10` and software writes the PE_CTRL.FLT_CFG to `0b01`, the value of PE_ST.FLT_CFG is updated to `0b10` without delay.

Changing terminated transaction response type

When a transaction is terminated by the Firewall Component, its configuration specifies whether it generates an error response or not. Software uses the PE_CTRL.ERR field to control this behavior.

The PE_ST.ERR field indicates the current response type to terminated transactions generated by the Firewall Component. When PE_CTRL.ERR and PE_ST.ERR are different, the Firewall Component is currently changing its terminated transaction response behavior.

The value of PE_ST.ERR only updates to the value of PE_CTRL.ERR, when any new transactions terminated by the Firewall Component are treated as per the value configured in the PE_CTRL.ERR field. Any transaction which has been terminated continues to behave as defined by the value of PE_ST.ERR at the point it failed the checks. It is UNPREDICTABLE whether a transaction which enters the Faulted state, while the values of PE_CTRL.ERR and PE_ST.ERR are different, uses the new or old value of PE_ST.ERR.

If software attempts to change the value of PE_CTRL.ERR, back to its previous value, when it differs from PE_ST.ERR, the Firewall treats the update as a Configuration Access Error and does not update the PE_CTRL register.

Arm strongly recommends:

- When software changes the value of the PE_CTRL.ERR field, it waits for the value of PE_ST.ERR to reflect this change before changing the value again.
- When software changes the values of PE_CTRL.ERR there are no outstanding transactions to the Firewall Component. The method software uses to achieve this is system-dependent. If there are outstanding transactions it is UNPREDICTABLE whether transactions terminated by the Firewall Component, use the new or old value of the PE_ST.ERR field.

Changing terminated transaction read data response

When a read transaction is terminated by the Firewall Component, it is configurable whether the read data is set to all 0s or a value dependent on the StreamID of the transaction.

Software uses the PE_CTRL.RAZ field to control this behavior. The PE_ST.RAZ field indicates the current value which the Firewall Component sets the read data to for terminated transactions. When PE_CTRL.RAZ and PE_ST.RAZ are different the Firewall Component is changing its terminated read response behavior.

The value of PE_ST.RAZ only updates to the value of PE_CTRL.RAZ, when any new read transactions terminated by the Firewall Component are treated as per the value configured in the PE_CTRL.RAZ field. Any transaction which has previously been terminated continues to behave as defined by the value of PE_ST.RAZ at the point it failed the checks. It is UNPREDICTABLE whether a read transaction which enters the Faulted state, while the values of PE_CTRL.RAZ and PE_ST.RAZ are different, uses the new or old value of PE_ST.RAZ.

If software attempts to change the value of PE_CTRL.RAZ, back to its previous value, when it differs from PE_ST.RAZ, the Firewall treats the update as a Configuration Access Error and does not update the PE_CTRL register.

Arm strongly recommends:

- When software changes the value of PE_CTRL.RAZ, it waits for the value of PE_ST.RAZ to reflect this change, before changing the value again.
- When software changes the values of PE_CTRL.RAZ there are no outstanding transactions to the Firewall Component. The method software uses to achieve this is system-dependent. If there are outstanding transactions it is UNPREDICTABLE whether read transactions terminated by the Firewall Component use the new or old value of PE_ST.RAZ field.

Enabling and disabling regions

The regions of the Firewall Component can be enabled and disabled using the RGN_CTRL0.EN field.

The status of the region is indicated by the RGN_ST.EN field. When RGN_CTRL0.EN and RGN_ST.EN have different values, it indicates that the Firewall Component is either performing an enablement or disablement of the region:

- Enablement: RGN_CTRL0.EN is 1 and RGN_ST.EN is 0
- Disablement: RGN_CTRL0.EN is 0 and RGN_ST.EN is 1

The value of RGN_ST.EN only updates, to the value in the RGN_CTRL0.EN field when the enablement or disablement process completes.

- The enablement process completes when the write to RGN_CTRL0.EN completes.
- The disablement process completes when no new transactions lookup against the region.

It is UNPREDICTABLE whether a transaction matches against the region, while the values of RGN_CTRL0.EN and RGN_ST.EN are different.

If software attempts to set the RGN_CTRL0.EN field, back to its previous value while an enablement or disablement is in-progress, the Firewall treats the update as a Configuration Access Error and does not update the RGN_CTRL0 register.

Arm strongly recommends:

- When software sets the value of RGN_CTRL0.EN, it waits for the value to be reflected in RGN_ST.EN before changing the value again.
- Before enabling the region, software waits for any previous configuration of the region to complete. If software enables the region while a previous configuration of the region is pending it is UNPREDICTABLE whether transactions, which are processed by the Firewall Component, see the new or old configuration values.
- Software waits for the disablement process to complete before attempting any other configuration of the region.
- Software waits for the enablement process to complete before issuing or allowing to be issued, any transactions which must match against the region.
- Software makes sure there are no outstanding and does not allow any transactions to be issued, which can match against the region being disabled.

Enabling and disabling MPEs

The MPEs of a region can be enabled and disabled using the RGN_CTRL1.MPE_EN{0-3} fields.

The status of the MPE is indicated by the RGN_ST.MPE_EN{0-3} fields. When RGN_CTRL1.MPE_EN{0-3} and RGN_ST.MPE_EN{0-3} have different values, it indicates that the Firewall Component is either performing an enablement or disablement of the MPE:

- Enablement: RGN_CTRL1.MPE_EN{0-3} is 1 and RGN_ST.MPE_EN{0-3} is 0
- Disablement: RGN_CTRL1.MPE_EN{0-3} is 0 and RGN_ST.MPE_EN{0-3} is 1

The value of RGN_ST.MPE_EN{0-3} only updates, to the value in the RGN_CTRL1.MPE_EN{0-3} field, when the enablement or disablement process completes. The enablement process completes when the write to RGN_CTRL1.MPE_EN{0-3} completes. The disablement process completes when no new transactions will lookup against the MPE. It is UNPREDICTABLE whether a transaction matches against the MPE, while the values of RGN_CTRL1.MPE_EN{0-3} and RGN_ST.MPE_EN{0-3} are different.

If software attempts to set the RGN_CTRL1.MPE_EN{0-3} field back to its previous value, while an enablement or disablement is in-progress, the Firewall treats the update as a Configuration Access Error and does not update any field in the RGN_CTRL1 register.

Arm strongly recommends:

- When software sets the value of RGN_CTRL1.MPE_EN{0-3}, it waits for the value to be reflected in RGN_ST.MPE_EN{0-3} before changing the value again.
- Software waits for the disablement process to complete before attempting to update the RGN_MID{0-3} and RGN_MPL{0-3} registers associated with the MPE.
- Software waits for the enablement process to complete before issuing or allowing to be issued, any transaction which match against the MPE.
- Software makes sure there are no outstanding transactions, and does not allow any transactions to be issued, which can match against the MPE being disabled.

Changing bypass mask

The Firewall Component is configured to use the value of its Bypass interface or not, for calculating whether it has been bypassed or not.

Software configures this using the PE_CTRL.BYPASS_MSK field. On writing this field, the PE_ST.BYPASS_MSK field is updated automatically to match the value written to the PE_CTRL.BYPASS_MSK field. The Firewall Component uses the value in the PE_ST.BYPASS_MSK to determine the use of the Bypass interface.

Changing bypass

The Firewall Component's protection logic is bypassed using the Bypass interface and the PE_ST.BYPASS_MSK.

The PE_BPS.BYPASS_ST field shows if the Firewall Component's protection logic is bypassed. The value of the field can change if ther is:

- A change in the value of PE_ST.BYPASS_MSK field
- A change in the value of the Firewall Component's Bypass interface, shown in the PE_BPS.BYPASS_IF_ST field

The value of the PE_BPS.BYPASS_ST only updates due to one of the above causes, when it is guaranteed that any new transactions processed by the Firewall Component's protection logic are treated as per the updated value of PE_BPS.BYPASS_ST.

Arm strongly recommends:

- Only changes PE_CTRL.BYPASS_MSK or the Bypass interface value at once.
- When software changes the value of PE_CTRL.BYPASS_MSK from 0 to 1, it waits for PE_BPS.BYPASS_ST to become 0 before changing the value of PE_CTRL.BYPASS_MSK again.
- When changing either the value of PE_CTRL.BYPASS_MSK or the Bypass interface, there are no outstanding transactions to the Firewall Component, as it is UNPREDICTABLE whether the transactions see the new or old value of the PE_BPS.BYPASS_ST field.
- Software only sets PE_CTRL.BYPASS_MSK to 1, when there are no requirements to perform SoC debug without software co-operation.

C.4.9 Protection logic terminated transaction response

The following table shows the type of response a Firewall Component's protection logic generates when it terminates a transaction which failed the checks and the protection logic.

Protection logic termination response

Transaction Type	PE_ST.ERR	PE_ST.RAZ	Response
Read	0	0	A non-error read response is generated with the read data set to a value specific to the StreamID
	0	1	A non-error read response is generated with the read data set to all 0s
	1	0	An error read response is generated with the read data set to a value specific to the StreamID
	1	1	An error read response is generated with the read data set to all 0s
Write	0	X	A non-error write response is generated
	1	X	An error write response is generated

The StreamID specific read data is IMPLEMENTATION DEFINED.

It is IMPLEMENTATION DEFINED whether a Firewall Component indicates whether the transaction has been terminated by the Firewall Component, or by the slave.

C.5 Monitor Extension (ME)

This section describes the ME.

When a Firewall Component implements ME.1 or greater, it includes monitor logic to detect errors generated by transactions issued by Masters connected to the Firewall Component.

Note

In this section any references to a Firewall Component, implicitly imply that the Firewall Component implements ME.1 or greater.

When a Firewall Component's monitor logic is enabled, it checks the transaction responses for errors and generate an error detection report with an Error Detection interrupt. For read transactions the read data can be either:

- Forwarded unmodified
- Replaced with a value specified by the StreamID of the master which issued the transaction

When a Firewall Component's monitor logic, is disable, the transaction responses are passed through to the issuing master without altering the read data or generating an error detection report and interrupt.

The monitor logic, can ignore transaction responses, using an IMPLEMENTATION DEFINED method, where the transaction response has:

- Been generated by another Firewall Component's protection logic terminating the transaction
- Already been logged by another Firewall Component's monitor logic

For more information see section *C.5.3 Error response ignore* on page Appx-C-346.

A Firewall Component must implement at least 1 error detection report. Software accesses the individual error detection reports using the Error Detection Window (EDW). For more information on error detection report and Error Detection Window, see *C.5.5 Registers* on page Appx-C-349.

This section contains the following subsections:

C.5.1 Error detection

The monitor logic checks the responses, on the Bus Master interfaces of the Firewall Component.

If the response indicates an error, the Firewall Component either:

- Logs the transaction and forwards the response to the issuing master. When the response is forwarded, for a read transaction the read data can be either set to a specific value or left unmodified.
- Forwards the response, without logging or modifying the read data.

Bus Protocol Error Response

Depending on the bus protocol used, the responses received may be either:

- Single cycle, for example, AXI for a write transaction
- Multi-cycle, for example, AXI for a read transaction

Note

When referring to multi-cycle responses, this does not include any cycles where the response is being stalled, for example in AXI the RREADY is low.

Some bus protocols support treating responses to transaction as either single or multi-cycle responses. For example, AHB can be considered as either. Each beat of the request has an associated response beat, except where the master terminates the burst early due to an error in a previous beat of the burst. The monitoring logic can treat these types of protocol as either single or multi-cycle, depending on the following rules:

- If the Firewall Component implements PE.1, or greater, if the protection checks are:
  - Applied on the entire transaction, the monitoring logic treats the response as multi-cycle
  - Applied on each beat of the transaction, the monitoring logic treats the response as a single cycle

In either case the information reported in the error detection report, for the address and transaction properties must be the same as the information reported in a fault entry, if the transaction failed the checks.

If the Firewall Component implements PE.0, it is IMPLEMENTATION DEFINED, whether it is treated as single or multi-cycle.

For multi-cycle responses the monitoring logic, implements a simple OR function for detection of errors. If any cycle of a response generates an error, then the whole of the response is considered to be an error and is reported. The address, reported in the error detection report, when ME.2 is implemented, depends on the bus protocol type:

- Burst-based protocols: Either:
  - Value of the beat which generated the error
  - Value provided at the start of the burst
- Beat-based protocols: Value provided of the beat, which generated the error.

Note

Arm recommends for burst-based the value provided is the value at the start of the transaction.

C.5.2 Error detection report

An error detection report contains the information about the transaction which generated the error response.

An error detection report can be:

- Valid: contains information about a transaction which caused an error response, which has not been acknowledge by software
- Invalid: contains an UNKNOWN value

Note

All error detection reports start in the invalid state, then transition into valid state when an error detection report details are loaded. Once software acknowledges the error detection report, it transitions into invalid state. After this the error detection report transitions between valid and invalid states on error response detection and software acknowledgement respectively.

Error detection reports are not accessed directly by software, but are instead accessed using the EDW. Only valid error detection reports are accessible in the EDW.

For information about how software can use error detection reports see section *C.12.2 Error detection report usage* on page Appx-C-394.

Error detection report properties

An error detection report contains the following information:

- MasterID
- Privilege level
- Data or instruction
- Security
- Read or write
- Address of transaction, when ME.2 is implemented

The information contained in the error detection report is of the incoming transaction and is not affected by any translation applied by the Firewall Component detecting the error.

Error detection report generation

When the monitor logic detects an error response, if::

- All error detection reports are valid then the monitor logic:
  - Generates an Error Detection Overflow interrupt
  - Does not generate the error detection report
  - Issues the response to the master which issued the transaction
- At least one error detection report is invalid then the monitor logic:
  - Generates the error detection report
  - Generates the Error Detection Interrupt
  - Issues the response to the master which issued the transaction

Note

For more information on how the monitor logic issues the response to the master see *Monitor Extension Control (ME_CTRL)* on page Appx-C-350

Error Detection Report Acknowledgement

Overview of error detection report acknowledgement.

An error detection report remains valid until software acknowledges the error detection report.

Error detection report overflow

It is possible for a Firewall Component to receive more error responses, than it has error detection reports.

When this occurs the monitor logic generates an Error Detection Overflow interrupt.

Error detection report and power

Error detection reports are lost when the Firewall Component enters the Disconnected state.

Software can use the ME_CTRL.EDR_PWR to set the ME_ST.EDR_PWR field and prevent Firewall Components from entering the Disconnected state when the EDW contains a valid error detection report.

Error detection window behavior

The EDW behaves as a FIFO with error detection reports automatically added when they are generated, provided an invalid error detection report exists.

When software acknowledges an error detection report, it is removed from the FIFO and the EDW then points to the next valid error detection report if one exists. If no more valid error detection report exists, the EDW:

- Indicates that there are no more valid error detection reports, by setting EDR_CTRL.EDR_VLD bit to 0
- EDR_TAL, EDR_TAU, EDR_TP and EDR_MID all read as 0x0
- Writes to the EDR_CTRL.ACK field are ignored

The EDR_CTRL register implements a field, EDR_CTRL.LAST_EDR, which indicates if the current error detection report is the last one in the EDW. When the EDW points to the last report this field reads as 1, otherwise it reads as 0.

C.5.3 Error response ignore

This section describes when an error response could be ignored.

The monitor logic can use an IMPLEMENTATION DEFINED method to ignore error responses, which have either:

- Been generated by another Firewall Component's protection logic, as part of a transaction being terminated
- Already been logged by another Firewall Component's monitor logic

This reduces the number of locations where either a terminated transaction or error response is logged, when Firewall Components are connected in parallel, as the following figure shows.

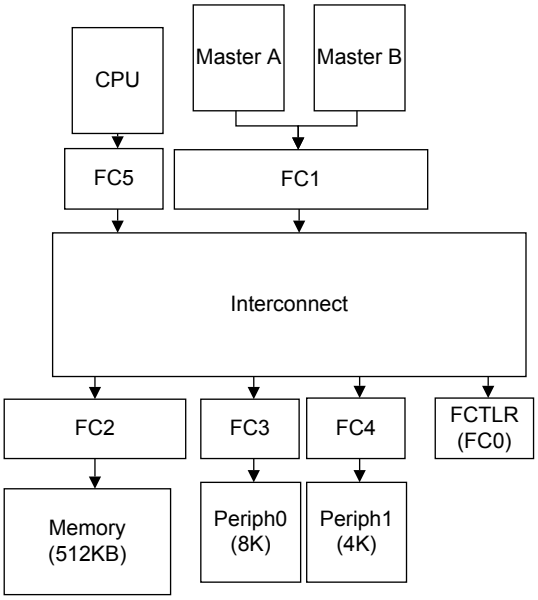

Figure C-7 Example system with Firewall Components both sides of the interconnect

Only FC1 and FC5 have ME.2 implemented. All other FCs have ME.0 implemented. When any of the following happens, the transaction is logged in the following location:

- For a transaction terminated by FC1, a fault entry is generated in the FC1.
- For a transaction terminated by FC{0,2-4}, a fault entry is generated in the Firewall Component which terminated the transaction. The response will also pass through either FC1 or FC5, depending on the master which issued the transaction. The Firewall component that terminated the transaction will use its IMPLEMENTATION DEFINED method to indicate that a fault has occurred. The error response should be ignored by other Firewall Components which the responses passes through and not generate an error detection report.

- For an error generated by either the memory or Periph 0 or 1, an error detection report is generated in the Firewall Component, FC1 or FC5, depending on the master which issued the transaction.
- For an error generated by FC0, an error detection report is generated in the Firewall Component, FC1 or FC5, depending on the master which issued the transaction.

——— **Note** ———

The above presumes the following:

- A Firewall Component, implementing PE.1 or greater, is configured to generate fault entries and error detection reports.
- A Firewall Component, implementing ME.1 or greater, is configured with its monitor logic enabled.
- The Firewall Controller is configured to generate an error for any configuration access which generates a Configuration Access Error.

———————————

C.5.4 Response processing

This section describes response processing sequence by a Firewall component.

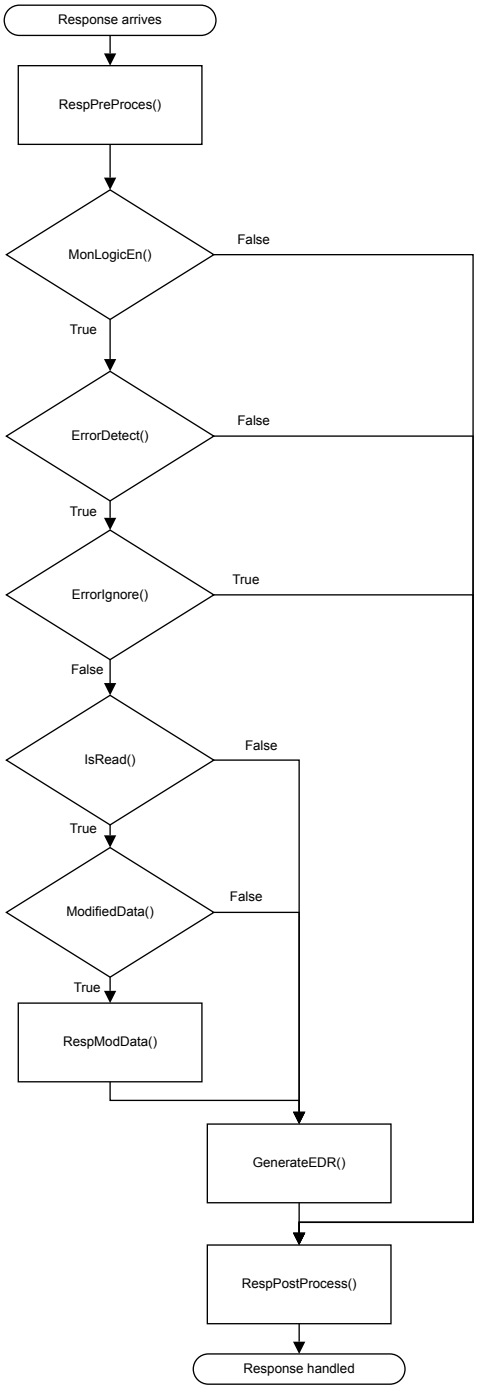

Figure C-8 Monitor logic transaction response processing flow

The following table gives a brief description of each function.

Table C-42 Response processing functions

Function	Description
RespPreProcess()	IMPLEMENTATION DEFINED function to convert the incoming bus protocol into the format used by the monitor logic. This function can block waiting for a new transaction.
RespPostProcess()	IMPLEMENTATION DEFINED function to covert the transaction format used by the monitor logic, into the format used by the Bus Slave interface.

Table C-42 Response processing functions (continued)

Function	Description
MonLogicEn()	This function returns true if the monitor logic is enabled, otherwise it returns false.
ErrorDetect()	This function returns true if the transaction response indicates an error, otherwise it returns false.
ErrorIgnore()	This function returns true if the transaction response indicates that it has either: • Been generated by another Firewall Component's protection logic, as part of a transaction being terminated • Already been detected by another Firewall Component's monitor logic Otherwise, it returns false.
IsRead()	This function returns true if the transactions which generated the response was a read transaction, otherwise it returns false.
ModifiedData()	This function returns true if the monitor logic is configured to modify the read data value, based on the StreamID of the transaction. Otherwise it returns false.
RespModData()	This function replaces the read data, with a value associated with the StreamID of the transaction.
GenerateEDR()	This function attempts to generate the error detection report and Error Detection interrupt, if there is an invalid error detection report in the EDW of the Firewall Component. Otherwise, it generates an Error Detection Overflow interrupt.

C.5.5 Registers

This section summarizes the registers which are defined by the Monitor Extension.

The Monitor Extension defines two types of registers:
- Monitor Control and Status
- Error Detection Window (EDW)

The Firewall Component's monitor logic can be either enabled or disabled by software. When the monitor logic is disabled, the Firewall Component, allows responses to pass through the Firewall Component unaltered. When the monitor logic is enabled it monitors the response to transactions.

A Firewall Component which implements ME.1 or greater, has the registers listed in the table below. These registers allow software to configure the monitor logic and access any error detection reports which have been generated. When ME.0 is implemented all these registers are Reserved and generate a Configuration Access Error when accessed.

Table C-43 Summary of Monitor Extension registers

Offset	Short name	Access	name
Control and Status			
0x200	*ME_CTRL* on page Appx-C-350	RW	Monitor Extension Control
0x204	*ME_ST* on page Appx-C-350	RO	Monitor Extension Status
Error Detection Window (EDW)			
0x260	*EDR_TAL* on page Appx-C-351	RO	Error Detection Report Transaction Address Lower
0x264	*EDR_TAU* on page Appx-C-351	RO	Error Detection Report Transaction Address Upper
0x268	*EDR_TP* on page Appx-C-351	RO	Error Detection Report Transaction Properties
0x26C	*EDR_MID>* on page Appx-C-352	RO	Error Detection Report MasterID
0x270	*EDR_CTRL* on page Appx-C-352	RW	Error Detection Report Control

The EDR_TAL and EDR_TAU registers are only implemented when ME.2 is implemented, otherwise the registers are Reserved.

Monitor Extension Control (ME_CTRL)

The following table gives a bit-level description of the Monitor Extension Control (ME_CTRL) register.

The Monitor Extension Control register allows software to enable the Firewall Component's monitor logic to monitor transactions responses and select the behavior of the monitor logic when an error is detected.

Table C-44 ME_CTRL register

Bits	Name	Description	Type	Reset
[31]	EN	Request the Firewall Component's monitor logic enables or disables. `0b0`: Request to disable the Firewall Component's monitor logic `0b1`: Request to enable the Firewall Component's monitor logic	RW	`0x0`
[30:5]	-	Reserved	RO	`0x000_0000`
[4]	EDR_PWR	Request error detection reports power behavior. `0b0`: Error detection report does not prevent entry into a Disconnected state `0b1`: Error detection report does prevent entry into a Disconnected state	RW	`0x0`
[3:2]	-	Reserved	RO	`0x00`
[1]	RDUM	Request behavior for read data returned for a read transaction which has caused an error. `0b0`: Read data is based on the StreamID `0b1`: Read data is left unmodified	RW	`0x0`
[0]	-	Reserved	RO	`0x0`

Arm recommends that when the value of the ME_CTRL.RDUM is changed there are no outstanding transactions. The method which software uses to guarantee this is outside the scope of Appendix C.

Monitor Extension Status (ME_ST)

The following table gives a bit-level description of the Monitor Extension Status (ME_ST) register.

Table C-45 ME_ST register

Bits	Name	Description	Type	Reset
[31]	EN	Status of the Firewall Component's monitor logic enables or disables. `0b0`: Firewall Component's monitor logic is disabled. `0b1`: Firewall Component's monitor logic is enabled. When SRE.1 is implemented and the Firewall Component has entered the Disconnected state this field matches the value in ME_CTRL.EN. As if the request to change from enabled to disable or disabled to enabled has completed.	RO	`0x0`
[30:5]	-	Reserved	RO	`0x000_0000`

Table C-45 ME_ST register (continued)

Bits	Name	Description	Type	Reset
[4]	EDR_PWR	Error detection reports power behavior. `0b0`: Error detection report does not prevent entry into a Disconnected state. `0b1`: Error detection report does prevent entry into a Disconnected state.	RO	0
[3:2]	-	Reserved	RO	`0x00`
[1]	RDUM	Request behavior for read data returned for a read transaction which has caused an error. `0b0`: Read data is based on the StreamID `0b1`: Read data is left unmodified	RO	0
[0]	-	Reserved	RO	0

See section *C.5.7 Monitor Logic Response Forwarding* on page Appx-C-355 for more information on how transaction responses are forwarded to the issuing master when the Firewall Component's monitor logic detects an error.

Error Detection Report Transaction Address Lower (EDR_TAL)

The following table gives a bit-level description of the Error Detection Report Transaction Address Lower (EDR_TAL) register.

This register is only implemented when ME.2 is implemented, otherwise it is Reserved and generates a Configuration Access Error when accessed.

Table C-46 EDR_TAL register

Bits	Name	Description	Type	Reset
[31:0]	ERROR_ADDR_LWR	Error transaction address lower. This field is RAZ when EDR_CTRL.EDR_VLD is 0.	RO	UNKNOWN

Error Detection Report Transaction Address Upper (EDR_TAU)

The following table gives a bit-level description of the Error Detection Report Transaction Address Upper (EDR_TAU) register.

This register is only implemented when ME.2 is implemented, otherwise it is Reserved and generates a Configuration Access Error when accessed.

Table C-47 EDR_TAU register

Bits	Name	Description	Type	Reset
[31:0]	ERROR_ADDR_UPR	Error transaction address upper. This field is RAZ when EDR_CTRL.EDR_VLD is 0.	RO	UNKNOWN

Error Detection Transaction Properties (EDR_TP)

The following table gives a bit-level description of the Error Detection Transaction Properties (EDR_TP) register.

Table C-48 EDR_TP register

Bits	Name	Description	Type	Reset
[31:22]	-	Reserved	RO	0x0000
[21]	W	Indicates whether the transaction, which caused the error, was a read or write. 0b0: Read 0b1: Write	RO	UNKNOWN
[20:19]	-	Reserved	RO	0x00
[18]	INST	Indicates whether the transaction, which caused the error, was an instruction or data access. 0b0: Data 0b1: Instruction	RO	UNKNOWN
[17]	PRIV	Indicates the privileged level of the transaction, which caused the error. 0b0: Unprivileged. 0b1: Privileged.	RO	UNKNOWN
[16]	NS	Indicates the security level of the transaction. 0b0: Secure 0b1: Non-secure	RO	UNKNOWN
[15:0]	-	Reserved	RO	0x00

Note

When EDR_CTRL.EDR_VLD is 0 this register reads as zero.

Error Detection Report MasterID (EDR_MID)

The following table gives a bit-level description of the Error Detection Report MasterID (EDR_MID) register.

Table C-49 EDR_MID register

Bits	Name	Description	Type	Reset
[31:0]	MST_ID	Indicates the MasterID of the transaction which caused the error. The width of this field depends on the value of FC_CFG2.MST_ID_WIDTH. Any unused bits are Reserved and treated as RAZ/WI.	RO	The reset value of this field depends on the value of FC_CFG2. SINGLE_MST: 0b0: UNKNOWN 0b1: Fixed valued defined at design time. When EDR_CTRL.EDR_VLD is 0 this register reads as zero.

Error Detection Report Control (EDR_CTRL)

The following table gives a bit-level description of the Error Detection Report Control (EDR_CTRL) register.

Table C-50 EDR_CTRL register

Bits	Name	Description	Type	Reset
[31]	LAST_EDR	Indicates if this error detection entry is the last valid entry. `0b0`: Report is not the last valid error detection report. `0b1`: Report is the last valid error detection report.	RO	`0b0`
[30]	EDR_VLD	Indicates whether the EDW is pointing to a valid error detection report. `0b0`: EDW is pointing to an invalid error detection report. `0b1`: EDW is pointing to a valid error detection report. When this field is `0b0` the values in the following registers read as 0: • EDR_TAL • EDR_TAU • EDR_TP • EDR_MID	RO	`0b0`
[29:1]	-	Reserved	RO	`0x0000_0000`
0	ACK	Acknowledge the error transaction. This field always reads as `0b0`. Writes to this field behave as follows: `0b0`: Ignored `0b1`: Error detection report is acknowledged. Writes to this register are ignored if EDR_CTRL.EDR_VLD is `0b0`.	WO	`0b0`

Software uses the EDR_CTRL to:
- Know if there is a valid error detection report in the EDW
- Acknowledge the current error detection report what EDW is currently pointing to
- Know if the current error detection report is the last one in the EDW

C.5.6 Changing configuration settings of monitor logic

This section describes the requirements for changing the configuration settings of the monitor logic.

Enabling and disabling monitor logic

The Firewall Component's monitor logic can be enabled and disabled using the ME_CTRL.EN field.

The status of the Firewall Component's monitor logic is indicated by the ME_ST.EN field. When ME_CTRL.EN and ME_ST.EN have different values, it indicates that the Firewall Component is either performing an enablement or disablement of the monitor logic:

- Enablement: ME_CTRL.EN is 1 and ME_ST.EN is 0
- Disablement: when ME_CTRL.EN is 0 and ME_ST.EN is 1

The value of ME_ST.EN only updates, to the value in the ME_CTRL.EN field, when the enablement or disablement process completes. The enablement process completes when the write to ME_CTRL.EN completes. The disablement process completes when any new transactions which complete with an error, do not generate an error detection report. It is UNPREDICTABLE whether a transaction which completes with an error, whilst the values of ME_CTRL.EN and ME_ST.EN are different, uses the old or new value of ME_ST.EN. This includes for multicycle responses, where the error response is received before disablement process starts, but the last cycle of the response is received after the disablement process started.

If software attempts to set the ME_CTRL.EN field, back to its previous value, whilst an enablement or disablement is in-progress, the Firewall treats the update as a Configuration Access Error and does not update the ME_CTRL register.

Arm strongly recommends:

- That when software has set the value of ME_CTRL.EN, it waits for the value to be reflected in ME_ST.EN before changing the value again.
- That before enabling the monitor logic, software waits for any previous configuration of the Firewall Component monitor logic to complete. If software enables the Firewall Component monitor logic, whilst a previous configuration of the Firewall Component monitor logic is pending, it is UNPREDICTABLE whether transactions which are processed by the Firewall Component use the new or old configuration values.
- Software waits for the disablement process to complete before attempting any other configuration of the Firewall Component's monitor logic. If software attempts to update the configuration of the Firewall Component's monitor logic, before the disablement process completes it is UNPREDICTABLE whether the processed transaction responses use the new or old configuration values.

Changing error detection report power behavior

The Firewall Component can be configured as to its behavior when the following conditions occur:

- Firewall Component has at least one valid error detection report
- Power Control interface, of the Firewall Component, makes a request to enter a power mode where the Firewall Component would be considered non-operational

Software can configure whether the Firewall Component accepts or denies the power request, under these conditions, using the ME_CTRL.EDR_PWR field. On writing this field the ME_ST.EDR_PWR field is updated automatically to match the value, written to the ME_CTRL.EDR_PWR field. The Firewall Component uses the value in the ME_ST.EDR_PWR to select whether it accepts or denies the power request.

Changing error transaction read data response

When the Firewall Component detects a response to a read transaction, which indicates an error has occurred, the Firewall Component can be configured whether to set the read data to all 0s or a value dependent on the StreamID of the transaction.

Software uses the ME_CTRL.RDUM field to configure this behavior. The ME_ST.RDUM field indicates how the Firewall Component will treat any response to a read transaction, which indicates an error has occurred. When ME_CTRL.RDUM and ME_ST.RDUM are different, the Firewall Component is changing its error transaction response behavior.

The value of ME_ST.RDUM only updates, to the value of ME_CTRL.RDUM, when any new response to a transaction will be treated as the value configured in the ME_CTRL.RDUM field. Any response which has previously been detected as an error behaves as defined by the value of ME_ST.RDUM at the point it was detected. It is UNPREDICTABLE whether an error read response, whilst the values of ME_CTRL.RDUM and ME_ST.RDUM are different, uses the old or new value of ME_ST.RDUM.

If software attempts to change the value of ME_CTRL.RDUM, back to its previous value, when it differs from ME_ST.RDUM, the Firewall treats the update as a Configuration Access Error and does not update the ME_CTRL register.

Arm strongly recommends:

- When software changes the value of either ME_CTRL.RDUM, it waits for the value of ME_ST.RDUM to reflect this change, before changing it again.
- When software changes the values of ME_CTRL.RDUM there are no outstanding transactions to the Firewall Component. The software method used to achieve this is system-dependent. If there are outstanding transactions it is UNPREDICTABLE whether a read transaction response, marked with an error, uses the new or old value of the ME_ST.RDUM field.

C.5.7 Monitor Logic Response Forwarding

The following table summarizes response forwarding in monitor logic.

Table C-51 Summary of response

Transaction type	ME_ST.EN	ME_ST.RDUM	Transaction response	Response
Read	0	X	X	Response is forwarded unmodified
	1	0	No error	Response is forwarded unmodified
			Error	Response is forwarded with the read data set to a value specific to the StreamID
	1	1	X	Response is forwarded unmodified
Write	X	X	X	Response is forwarded unmodified

The StreamID specific read data is set by an IMPLEMENTATION DEFINED method.

It is IMPLEMENTATION DEFINED whether a Firewall Component indicates it has logged the error or not.

This reduces the number of places that the transaction will be logged. For more information see section *C.5.3 Error response ignore* on page Appx-C-346.

C.6 Translation Extension

This section describes Translation Extension.

The Translation Extension level 1 or greater can only be implemented when PE.1 or greater is implemented.

The Translation Extension allows the Firewall Component to modify properties of the outgoing transaction.

Translation is only applied after a transaction has passed the checks and the region, used to perform the checks, was configured as a power of 2 in size (RGN_SIZE.MULnPO2 set to 0). The translation settings for the region, used to perform the checks, are used to generate the output transaction.

This section contains the following subsections:

C.6.1 Region Properties

The Translation Extension adds the following properties to a region:

- Output transaction translation address
- Output transaction memory attribute property
- Output transaction security property
- Output transaction instruction property
- Output transaction privilege level property

Some of the properties are dependent on:

- The level of extensions implemented
- The design time configuration of Firewall Component
- How software has programmed the Firewall Component

In *Output transaction translation address* on page Appx-C-356 to *Output transaction privilege level property* on page Appx-C-360 each property is covered in detail.

Output transaction translation address

A region can define an output transaction translation address, which the Firewall Component uses to modify the address of the output transaction.

The following properties are used to calculate the output transaction's address (OTA):

- Translation address (TA)
- Region size (RGN_SIZE)
- Incoming transaction's address (ITA), zero extended to 64 bits
- Firewall Component's Protection Size (PROT_SIZE)

The following formula shows how the OTA is calculated:

OTA<63:0> = ITA<63:PROT_SIZE>:TA<PROT_SIZE-1:RGN_SIZE>:ITA<RGN_SIZE-1:0>

The address translation is only applied when the address translation enable is set to 1, otherwise the output transaction's address is the same as the incoming transaction's address.

The translation address is only programmable when TE.2 is implemented and the RGN_SIZE.MULnPO2 is 0. Otherwise the output transaction's address is the same as the incoming transaction's address.

The table below shows when address translation is applied.

Table C-52 Behavior of address translation

Level of TE	RGN_SIZE.MULnPO2	RGN_TCFG2.ADDR_TRANS_EN	Translation applied
0	0	X	No
1		X	No
2		0	No
		1	Yes
X	1	X	No

Output transaction memory attribute property

This section describes how the output transaction memory attribute properties are determined.

A region defines an output transaction memory attribute property which is programmable. The table below shows the outgoing transaction memory attribute value depending on the configuration of the Firewall Component and the programming of the region.

Table C-53 Behavior of memory attribute property translation

Level of TE	FC_CFG1.MA_SPT	RGN_TCFG2. MA_TRANS_EN	RGN_SIZE. MULnPO2	Outgoing transaction memory attribute
0	X	X	0	If the bus protocol supports memory attribute, it remains unchanged from the incoming transaction
>=1	0	X		Bus protocol does not support memory attribute
	1	0		Outgoing transaction has the same memory attribute as the incoming transaction
		1		Outgoing transaction has the memory attributed defined by RGN_TCFG2.MA
X	X	X	1	Region is not a power of 2 so translation is not applied. Incoming memory attributes are used unchanged

Note

The outgoing transaction may have different values for the memory attribute property than defined in the table above.

The memory property is defined as an 8-bit value which defines the inner and outer memory type, cache allocation policy, and whether it is transient or not.

The table below contains the list all legal values and the memory type.

Table C-54 Memory types

7:4	3:0	Device or Normal	Outer Cacheability	Inner Cacheability
0b0000	0b0000	Device-nGnRnE	N/A	N/A
	0b0100	Device-nGnRE	N/A	N/A
	0b1000	Device-nGRE	N/A	N/A
	0b1100	Device-GRE	N/A	N/A
0b00RW, RW not 0b00	0b00RW, RW not 0b00	Normal	Write-Through Transient	Write-Through Transient
	0b0100			Non-cacheable
	0b01RW, RW not 0b00			Write-Back Transient
	0b10RW			Write-Through Non-transient
	0b11RW			Write-Back Non-transient
0b0100	0b00RW, RW not 0b00		Non-cacheable	Write-Through Transient
	0b0100			Non-cacheable
	0b01RW, RW not 0b00			Write-Back Transient
	0b10RW			Write-Through Non-transient
	0b11RW			Write-Back Non-transient
0b01RW, RW not 0b00	0b00RW, RW not 0b00		Write-Back Transient	Write-Through Transient
	0b0100			Non-cacheable
	0b01RW, RW not 0b00			Write-Back Transient
	0b10RW			Write-Through Non-transient
	0b11RW			Write-Back Non-transient
0b10RW	0b00RW, RW not 0b00		Write-Through Non-transient	Write-Through Transient
	0b0100			Non-cacheable
	0b01RW, RW not 0b00			Write-Back Transient
	0b10RW			Write-Through Non-transient
	0b11RW			Write-Back Non-transient
0b11RW	0b00RW, RW not 0b00		Write-Back Non-transient	Write-Through Transient
	0b0100			Non-cacheable
	0b01RW, RW not 0b00			Write-Back Transient
	0b10RW			Write-Through Non-transient
	0b11RW			Write-Back Non-transient

R indicates the read allocation policy. Bit 5 indicates the outer read allocation, whilst bit 1 indicates the inner.

W indicates the write allocation policy. Bit 4 indicates the outer write allocation, whilst bit 0 indicates the inner.

R or W:

- 0 – No allocate
- 1 – Allocate

Any values not listed in the table above are Reserved and behave as follows:

- When bits 7:4 == 0000, bits 1:0 are ignored and treated as 00. For example, a value of `0x0F` is treated as Device-GRE.
- When bits 7:4 != 0000, the memory is treated as Normal with the inner cache policy set to Write-Through Transient Read and Write allocate. The outer cache policy is set by bits 7:4. For example, a value of `0x70` is treated as Normal memory outer Write-Back Transient Read and Write allocate, inner Write-Through Transient Read and Write allocate.

Note

Arm strongly recommends software never sets the memory type to a Reserved value.

Output transaction security property

This section describes how the output transaction security property is determined.

A region defines an output transaction security property which is programmable. The table below shows the security of the outgoing transaction depending on the configuration of the Firewall Component and the programming of the region.

Table C-55 Behavior and legality of security property translation

Level of TE	Level of SE	FC_CFG0.SEC_SPT	Incoming transaction security	RGN_TCFG2.NS	RGN_SIZE.MULnPO2	Outgoing transaction security
0	X	X	X	X	`0b0`	If the bus protocol supports a security property it remains unchanged.
>=1	0	`0b0`	X	X		
	0	`0b1`	X	X		Illegal Firewall Component configuration.
	1	`0b0`	X	X		
		`0b1`	X	`0b00`		If the bus protocol supports a security property it remains unchanged.
		`0b1`	S	`0b10`		S
				`0b11`		NS
			NS	X		NS
X	X	X	X	X	`0b1`	Region is not a power of 2 so translation is not applied. Incoming security is used unchanged.

Output transaction instruction property

This section describes how the output transaction instruction property is determined.

A region defines an output transaction instruction property (whether the access is an instruction or data access) which is programmable. The table below shows the instruction property of the outgoing

transaction depending on the configuration of the Firewall Component and the programming of the region.

Table C-56 Behavior of instruction property translation

Level of TE	FC_CFG0.INST_SPT	RGN_TCFG2.INST	RGN_SIZE.MULnPO2	Outgoing transaction instruction property
0	X	X	0	If the bus protocol supports instruction property, it remains unchanged from the incoming transaction
>=1	0	X		Firewall Component does not support the property
	1	0b00		Incoming instruction property is used unchanged
		0b10		Data
		0b11		Instruction
X	X	X	1	Region is not a power of 2 so translation is not applied. Incoming instruction property is used unchanged.

Note

The translation of the instruction property only applies to read transactions. All write transactions are considered to be data accesses.

Output transaction privilege level property

This section describes how the output transaction privilege level property is determined.

A region defines an output transaction privilege level property (whether the access is from a privileged or unprivileged master), which is programmable. The table below shows the privilege of the outgoing transaction depending on the configuration of the Firewall Component and the programming of the region.

Table C-57 Behavior of privilege level property translation

Level of TE	FC_CFG0.PRIV_SPT	RGN_TCFG2.PRIV	RGN_SIZE.MULnPO2	Outgoing transaction privilege property
0	X	X	0b0	If the bus protocol supports privilege property, it remains unchanged from the incoming transaction
>=1	0b0	X	-	Firewall Component does not support the property
	0b1	0b00		Incoming privilege property is used unchanged
		0b10		Unprivileged
		0b11		Privileged
X	X	X	0b1	Region is not a power of 2 so translation is not applied. Incoming privilege property is used unchanged.

Translation enables

When TE.1 is implemented a region implements the following translation enables:

- Memory Attribute
- Instruction

- Privilege
- Shareability
- Security

When TE.2 is implemented a region also implements the address translation enable.

Translation of the properties are only applied when RGN_SIZE.MULnPO2 is 0. Otherwise the outgoing transaction has the same values as the incoming transaction, irrespective of the value of the translation enable.

C.6.2 Property translation

When the Firewall Component supports TE.1 or greater, it can modify the following properties of the outgoing transaction:ion property translation.

- Shareability
- Memory attribute
- Security (for a Secure transaction only)
- Instruction
- Privilege

Software can select, for each property, to use either the incoming value, if the value is not supplied on the incoming Bus Slave interface the default value is used instead, or a software defined value.

It is possible for outgoing transaction to have illegal combination of properties, due to a combination of software programming and incoming values.

As it is possible for software to perform translation on any power of 2 size granule, it is possible for two different masters to access the memory with different attributes. This can cause visibility and coherency issues. Software is responsible for making sure that the memory attribute properties used by the different masters are correct and if any cache maintenance operations are required to make sure that updates made by one master are visible to another.

C.6.3 Address translation

When the Firewall Component supports TE.2, it can modify part of the address of the transaction.

Note

The Translation Extension of the Firewall is not a replacement for a *Memory Management Unit* (MMU).

The Firewall considers the incoming address of the transaction as the virtual address of the transaction, and the output transaction as the physical or intermediate physical address, just like an MMU. The difference between the Firewall and an MMU is the range of address that can be translated across, and the number of translations that can be configured. An MMU allows any incoming address to map to any outgoing address, and in some cases allows a larger outgoing address range than an incoming address range. The Firewall defines the range of output addresses a transaction can target based on the Protection Size of the Firewall Component. It is only possible to translate transactions within the region of the address space which a region can be defined in.

The range of output addresses is limited by the Protection Size of the Firewall Component. Only the bits, defined by the Protection Size of the Firewall Component are translated. For more information on how address translation is performed see *Output transaction translation address* on page Appx-C-356.

Address translation and protection extension level

For a Firewall Component which supports PE.1 the incoming address range for each region is fixed at design time. Therefore, software is only able to define the output address.

For a Firewall Component which supports PE.2 software can define the input and output address for a region.

The following figure shows the difference between a Firewall Component, implementing TE.2 and PE.1 against implementing TE.2 and PE.2.

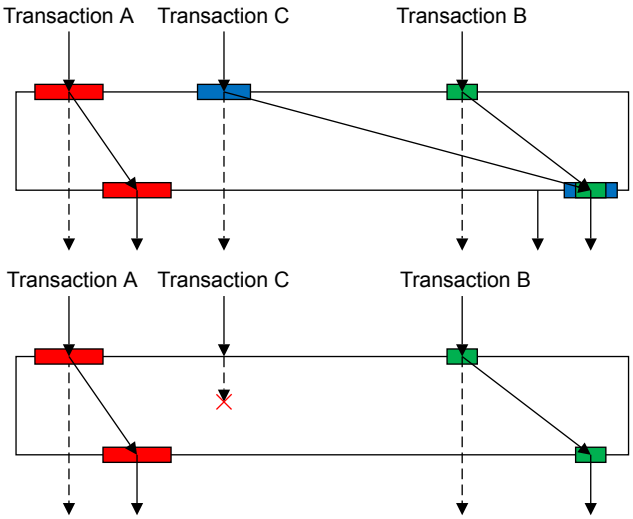

Figure C-9 Difference between a Firewall Component, implementing PE.1 or PE.2, for address translation

Both Firewall Components have two regions defined. The red region allows Transaction A to be remapped in both cases from the input to output address as the figure shows. The green region allows Transaction B to be remapped in both cases from the input to the output address. When Transaction C arrives, it is required to define a new region. For the Firewall Component, implementing PE.2, software defines the new region, blue, with the input and output address. In this example, Transaction C is mapped to the same location as Transaction B. But for a Firewall Component, implementing PE.1, software cannot define new regions and instead is only able to remap regions which have been predefined. This means that Transaction C will always cause a fault and therefore the transaction must be terminated.

In the following figure, the Protection Size of the Firewall Components is defined to match the address range of the incoming and output bus fabric. However, it is allowed for the Protection Size to be smaller. This can be either at design time or changed using the Protection Size interface. For example, the following figure shows the Protection Size of the Firewall Component is cut in half. This causes the following:

- Transaction B will always fail the protection logic checks.
- Transaction C will translate to a different address.

In the following figure, the dotted lines show the original Protection Size and translation. The solid lines show the translation when the Protection Size has been halved. For Transaction A the translation is the same in both cases.

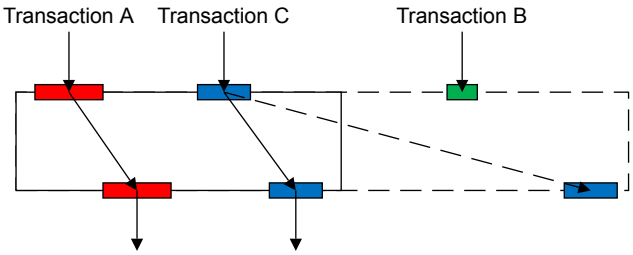

Figure C-10 Example of a PE.2 Firewall Component where the Protection Size has been halved

C.6.4 Registers

The Translation Extension adds the RGN_TCFG{0-2} registers.

Region Translation Config {0,1} (RGN_TCFG{0,1}) on page Appx-C-330 to *Region Translation Config 2 (RGN_TCFG2)* on page Appx-C-331 describe these registers. When TE.1 is implemented RGN_TCFG2 is the only register implemented, with RGN_TCFG{0-1} Reserved.

When TE.2 is implemented RGN_TCFG{0-2} are all implemented.

Note

RGN_TCFG{0-1} are also implemented if RSE.1 is implemented. See *C.7 Region Size Extension* on page Appx-C-364 for more information on the RSE.

C.7 Region Size Extension

The Region Size Extension level 1 can only be implemented when PE.1 or greater is also implemented by the Firewall Component.

The Region Size Extension allows regions to have a size which is not equal to a power of 2. This allows regions to be defined to better match the area of memory it is protecting. For example, if there was a buffer in memory 12KB in size. When RSE.0 is implemented the region is defined as 16KB, this means that the memory between 12KB and 16KB is not able to be used for another device. However, when RSE.1 is implemented the region can be defined to be 12KB and no memory is wasted.

Note

Translation is only applied if the RGN_SIZE.MULnPO2 is 0.

This section contains the following subsection:

- *C.7.1 Registers* on page Appx-C-364.

C.7.1 Registers

The Region Size Extension adds the following registers and fields:

- RGN_TCFG{0-1} registers, if not already implemented by the Translation Extension
- RGN_SIZE.MULnPO2

For more information on the RGN_TCFG{0-1} register see *Region Translation Config {0,1} (RGN_TCFG{0,1})* on page Appx-C-330 and *Region MasterID {0-3} (RGN_MID{0-3})* on page Appx-C-332. For more information on the RGN_SIZE.MULnPO2, see *Region Size (RGN_SIZE)* on page Appx-C-329.

C.8 Security Extension

The Security Extension is implemented at the Firewall level, with all Firewall Components implementing the same features.

A Firewall, which implements SE.0 does not consider the security property of transactions it is processing or monitoring.

——— **Note** ———

Arm strongly recommends that SE.0 is only implemented in a system that does not support Arm TrustZone.

———————————

A Firewall, which implements SE.1 has Firewall Components which:

- Performs checks on the security of the transaction when PE.1, or greater, is implemented.
- Reports the security of a transaction, which fails the checks when PE.1, or greater, is implemented.
- Reports the security of a transaction, which caused an error response when ME.1, or greater, is implemented.
- Translating the security of a transaction, from Secure to Non-secure, when TE.1, or greater, is implemented.

C.9 Lockdown Extension

This section describes the Lockdown Extension.

The Lockdown Extension is implemented at the Firewall level, with all Firewall Components within the same Firewall implementing the same features. The Lockdown Extension allows registers of the Firewall Component to be prevented from being updated. Depending on the level of the extension implemented, different granularities of lockdown can be applied.

All Firewall Components have a lockdown state, as defined in *C.9.1 Firewall Component lockdown* on page Appx-C-366. Depending on the level of LDE implemented defines the lockdown states the Firewall Component can enter:

- LDE.0: Always in Open lockdown state
- LDE.1: Open or Full
- LDE.2: Open, Partial or Full

Alongside the Firewall Component lockdown state, when LDE.2 is implemented, each region has a lockdown state: Unlocked or Locked.

When a Firewall Component is in the Partial or Full lockdown state or a region is in the Locked state, a configuration access which attempts to update certain registers behave as follows:

- Configuration Access Error is generated.
- Tamper interrupt and Tamper report is generated, if there is not a valid Tamper report present. Otherwise a Tamper Overflow interrupt is generated.

This section contains the following subsections:

C.9.1 Firewall Component lockdown

This section describes which registers and fields can be updated in the different lockdown states.

Open

All registers can be updated, except for RWE registers, when the RWE_CTRL points to a region which is in the Locked state.

For more information on region lockdown see *C.9.2 Region lockdown* on page Appx-C-367.

Partial

All registers, except for the following, are read-only:

- RWE_CTRL
- FE_CTRL
- EDR_CTRL
- RWE registers, when the RWE_CTRL points to a region which is in the Unlocked state. For more information on region lockdown see *C.9.2 Region lockdown* on page Appx-C-367.

The LD_CTRL register is read-only, when in the Partial lockdown state, and the LD_CTRL.LDI_ST of the Firewall Controller reads as 1.

Any attempt to:

- Update a register which is read-only due to the lockdown state of the Firewall Component
- Update the value of the LD_CTRL register when the Firewall Component is in Partial lockdown state and the LD_CTRL.LDI_ST of the Firewall Controller reads 1

Generates a:

- Configuration Access Error
- Tamper interrupt and Tamper report, if there is no valid Tamper report present. Otherwise, a Tamper Overflow interrupt is generated.

Note

This does not include registers which are Reserved due to the configuration of the Firewall Component or registers which are always read-only.

Full

All registers, except for the following, are read-only:

- RWE_CTRL
- FE_CTRL
- EDR_CTRL

The LD_CTRL register is read-only, when in the Full lockdown state, and the LD_CTRL.LDI_ST of the Firewall Controller reads 1.

Any attempt to:

- Update a register which is read-only due to the lockdown state of the Firewall Component
- Update the value of the LD_CTRL register when the Firewall Component is in Partial lockdown state and the LD_CTRL.LDI_ST of the Firewall Controller reads 1

Generates a:

- Configuration Access Error
- Tamper interrupt and Tamper report, if there is no valid Tamper report present. Otherwise, a Tamper Overflow interrupt is generated.

Note

This does not include registers which are Reserved due to the configuration of the Firewall Component or registers which are always read-only.

C.9.2 Region lockdown

When a region is locked, RGN_LCTRL.LOCK set to 1, the following registers are read-only:

- RGN_CTRL{0-1}
- RGN_CFG{0-1}
- RGN_SIZE
- RGN_TCFG{0-2}
- RGN_MID{0-3}
- RGN_MPL{0-3}

The RGN_LCTRL.LOCK field can be prevented from being updated by entering the Firewall Component into the Partial or Full lockdown state. When in the Partial lockdown state unlocked regions can still be programmed and locked, but locked regions cannot be programmed or have the RGN_LCTRL.LOCK field updated. When in the Full lockdown state, no region can be programmed and the RGN_LCTRL.LOCK field cannot be updated, irrespective of the lockdown state of the region.

Any attempt to:

- Update any of the above registers, which are implemented and not Reserved, when the RGN_LCTRL.LOCK is 1

Note

This also applies if the RGN_CTRL{0-1}, RGN_MID{0-3} or RGN_MPL{0-3} are read-only due to the configuration of the Firewall Component.

- Update the RGN_LCTRL.LOCK field when RGN_LCTRL.LOCK is 1 and the Firewall Component is in the Partial lockdown state
- Update the RGN_LCTRL.LOCK field when the Firewall Component is in the Full lockdown state

Generates a:

- Configuration Access Error
- Tamper interrupt and Tamper report, if there is no valid Tamper report present. Otherwise, a Tamper Overflow interrupt is generated.

Note

It is possible to lock a region when the region is not enabled.

For information on how software locks and unlocked regions see *Locking and unlocking regions* on page Appx-C-369.

C.9.3 Lockdown interface

When LDE.1, or greater, is implemented the Firewall Controller implements the Lockdown interface.

C.2.5 Lockdown interfaces on page Appx-C-294 defines the Lockdown interface.

C.9.4 Registers

This section describes the registers defined for Lockdown Extension.

Table C-58 Summary of Lockdown registers

Offset	Short Name	Access	Name
0x010	LD_CTRL	RW	Lockdown Control

Lockdown Control Register (LD_CTRL)

The following table gives a bit-level description of the Lockdown Control (LD_CTRL) register.

The Lockdown Control register, LD_CTRL, allows software to configure the lockdown state of the Firewall Component, see *C.9.1 Firewall Component lockdown* on page Appx-C-366 for more information on lockdown states of Firewall Components. The register is only implemented when LDE.1 or greater is supported by the Firewall, otherwise the register is Reserved and generates a Configuration Access Error when accessed.

Table C-59 LD_CTRL register

Bits	Name	Description	Type	Reset
[31:3]	-	Reserved	RO	`0x0000_0000`
[2]	LDI_ST	Lockdown interface status. Indicates the current value of the Lockdown interface. `0b0`: Lockdown interface is de-asserted. `0b1`: Lockdown interface is asserted. This field reads as `0b0` for Firewall Components other than the Firewall Controller. The value of this register is dependent on the value of the Lockdown interface of the Firewall.	RO	See description
[1:0]	LOCK	Indicates the lock state of the Firewall Component. `0b00`: Open lockdown state `0b01`: Reserved and treated as `0b00` `0b10`: Partial lockdown state. When LDE.1 is implemented this value is Reserved and treated as `0b00`. `0b11`: Full lockdown state	RW	`0x00`

The LD_CTRL.LOCK field is not updateable, when the following are all true:

- LD_CTRL.LDI_ST, of the Firewall Controller, is `0b1`
- Lockdown state of the Firewall Component is Partial or Full

Any attempt to update the LD_CTRL.LOCK field, under these conditions generates:

- A Configuration Access Error
- A Tamper interrupt and Tamper report, if there is no valid Tamper report present. Otherwise, a Tamper Overflow interrupt is generated.

C.9.5 Changing lockdown state of Firewall Component and regions

This section describes changing the lockdown state of Firewall Component and regions.

Changing lockdown state of Firewall Component

When software wants to change the lockdown state of the Firewall Component it writes to the LD_CTRL.LOCK field.

Arm recommends that software checks the value of the LD_CTRL.LDI_ST field, in the Firewall Controller, before attempting to update the LD_CTRL.LOCK field. Software must use a system-dependent manner to change the value of Lockdown interface.

Locking and unlocking regions

When LDE.2 is implemented regions of the Firewall Component can be locked using the RGN_LCTRL.LOCK field. Arm recommends software only locks a region after all other region programming is complete.

C.10 Save and Restore Extension

This section describes the details of Save and Restore Extension.

This section contains the following subsections:

C.10.1 Shadow Registers

Overview of shadow registers.

When the Firewall implements the SRE.1 extension it also contains shadow registers, which are used to preserve the state of certain registers when the associated Firewall Component enters the Disconnected state. The registers which are preserved are:

- PE_CTRL
- RWE_CTRL
- ME_CTRL
- LD_CTRL.LOCK
- FW_CTRL
- FC{0-31}_INT_MSK
- For each region independent of whether it is enabled or not:
  - RGN_CTRL{0-1}
  - RGN_LCTRL
  - RGN_CFG{0-1}
  - RGN_SIZE
  - RGN_TCFG{0-2}
  - RGN_MPL{0-3}
  - RGN_MID{0-3}
- Sampled value of all Protection Size interfaces, which are associated with a Firewall Component which implements PE.2

C.10.2 Register Behavior when SRE.0 Implemented

Description of register behavior when SRE.0 is implemented.

Register behavior when SRE.0 is implemented is as follows:

- When the component is in the Disconnected state, all accesses to its configuration registers return either:
  - A Configuration Access Error for a Firewall Component other than the Firewall Controller.
  - Enter the Pending state for the Firewall Controller.
- When the component is in the Connecting or Disconnecting state, the access is stalled until the component enters the Connected state.
- When the component is in the Connected state all accesses must complete as normal.

C.10.3 Register Behavior when SRE.1 Implemented

Description of register behavior when SRE.1 is implemented.

Register behavior when SRE.1 is implemented as follows:

- Accesses to the registers of the Firewall Controller, which is in the Disconnected state, all enter the Pending state.

Note

When the Firewall is integrated into a system it is possible for a deadlock to occur if an access is made to the Firewall whilst the Firewall Controller is in the Disconnected state.

- Accesses to the registers of Firewall Components other than the Firewall Controller, which is in the Disconnected state, behave as follows:
  - Read accesses to the following registers or field return the reset value:
    - FE_CTRL.{LAST_FE,FE_VLD}
    - EDR_CTRL.{LAST_EDR,EDR_VLD}
  - Read accesses to the following registers or field return 0:
    - PE_BPS
    - FE_TAL
    - FE_TAU
    - FE_TP
    - FE_MID
    - FE_CTRLACK
    - EDR_TAL
    - EDR_TAU
    - EDR_TP
    - EDR_MID
    - EDR_CTRL.ACK
  - Reads accesses to the following register or fields return a value dependent on another register:
    - PE_ST returns the value in the PE_CTRL register.
    - ME_ST returns the value in the ME_CTRL register.
    - RGN_ST.EN returns the value in RGN_CTRL0.EN field.
    - RGN_ST.MPE_EN{0-3} returns the value in RGN_CTRL1.MPE_EN{0-3} field.
  - Reads accesses to any other registers contained in the shadow registers, return the current value of the register in the shadow register.
  - Read accesses to any FC_CFG2.PROT_SIZE returns the value sampled on the Protection.
  - Read accesses to any registers not contained in the shadow registers, return their reset values.
  - Write accesses to the following registers or fields are ignored, but do not generate a Configuration Access Error:
    - FE_CTRL.ACK.
    - EDR_CTRL.ACK
  - Write accesses to the other registers or fields update the shadow register. The Firewall Controller is responsible for making sure that the most up-to-date value is restored when the Firewall Component enters the Connected state and before it starts processing transactions.
- When the component is in the Connecting state, this only applies to Firewall Components.
  - Reads and write accesses are stalled until the Firewall Component enters the Connected state.
- When the component is in the Connected state read and write accesses are as normal.
- When the component is in the Disconnecting state, this only applies to Firewall Components.
  - Read and write accesses are stalled until the Firewall Component enters the Disconnected state.

The above description is only for registers which are implemented by the Firewall Component, independent of the state of the Firewall Component. If a location which is Reserved due to level of extensions implemented by the Firewall Component, the access always generates a Configuration Access Error.

If a field within a register is Reserved, it always returns its Reserved value when the register is accessed independent of the state of the Firewall Component.

C.10.4 Shadow Register Initialization

Overview of shadow register initialization.

The shadow registers may need to be initialized before use. During the initialization period software can only access the following registers:

- Accesses to the FW_SR_CTRL register.
- Accesses to the FC_CAP{0-3} or FC_CFG{0-3} registers of all Firewall Components.
- Accesses to the Peripheral and Component Identification registers of the Firewall Controller.

——— **Note** ———

FC_CAP{0-3} and FC_CFG{0-3} are read-only registers. A write access to any of these registers remains unaffected by the shadow register initialization. For accesses to all other registers the Firewall generates a Configuration Access Error.

———————————

——— **Note** ———

Accesses to address locations within the Firewall which are Reserved, are unaffected by the shadow register initialization.

———————————

Whilst the shadow registers are being initialized, the FW_SR_CTRL.SR_RDY field reads as 0. FW_SR_CTRL.SR_RDY becomes 1 when the initialization is complete.

The FW_SR_CTRL.SR_RDY field will remain 1 unless the following occurs:

- Firewall Controller enters the Disconnected state and the shadow registers were not required to retain their values. For example, the Firewall Controller enters a non-operational mode when FW_SR_CTRL.SR_PWR is 0.

Arm recommends that software polls on the FW_SR_CTRL.SR_RDY field to become 1 before attempting to program the Firewall.

C.11 Firewall Controller

The Firewall Controller (FCLTR) is a modified Firewall Component, which provides the following features to the Firewall:

- Access control to the Firewall Components, within the Firewall
- Save and Restore functionality to the Firewall Components, if SRE.1 is implemented
- Lockdown interface, if LDE.1 or greater is implemented
- Interrupt interface
- Tamper Interrupt interface, if LDE.1 or greater is implemented
- Protection Size interfaces, for any Firewall Component which implements PE.2

The main task of the Firewall Controller is to implement Firewall Component 0. Firewall Component 0 provides access control to all Firewall Components in the Firewall. It is possible to provide access control to regions which are not part of the Firewall. However, care must be taken, in system integration and usage of the Firewall, not to affect the protection Firewall Component 0 provides to the Firewall.

Note

The terms Firewall Controller and Firewall Component 0 refer to the same thing in this document.

The Firewall Controller, like other Firewall Components, is allocated 64KB from the Firewall memory map. The Firewall Controller occupies the first 64KB of the memory allocated to the Firewall.

Sections *C.11 Firewall Controller* on page Appx-C-373 to *C.11.8 Changing Configurations Settings of Firewall* on page Appx-C-391 describe the difference between the Firewall Controller and any other Firewall Component.

This section contains the following subsections:

C.11.1 Protection Extension

The Firewall Controller always supports PE.1.

The Firewall Controller is implemented as described in *C.4 Protection Extension* on page Appx-C-305and except for the following differences:

- Reset value of the PE_CTRL.EN and PE_ST.EN is 1
- It always supports at least 3 regions
- Region 0 is enabled by default

Arm strongly recommends that software never sets the PE_CTRL.EN to 0 when the PE_ST.BYPASS_MSK is 1 or PE_BPS.BYPASS_ST is 0. If PE_CTRL.EN is set to 0 under either of those conditions, the Firewall Controller will become disabled and all transaction will be considered to fail the protection logic checks. Therefore, software will be unable to perform any other configuration accesses to the Firewall, once the Firewall Controller is disabled. The only way to reverse setting PE_CTRL.EN to 0 once PE_ST.EN is 0, is to enter the Firewall Controller into a Disconnected state before returning it back to the Connected state. The method by which software achieves this is outside the scope of *Appendix C Firewall* on page Appx-C-281.

Regions and RWE

The following table describes the regions defined for the Firewall Controller.

Table C-60 Region summary for Firewall Component 0

Region number	Description	Components
0	The region is enabled by default and all fields are read-only. The region has a pre-populated MPE which allows a single StreamID to access the Firewall. The agent allocated with this StreamID is considered the Configuration Master.	Firewall Controller
1	Standard region, except for the following fields, of the RWE, are always RAZ/WI: • Secure Privileged Execute (RGN_MPL{0-3}.SPX) • Secure Unprivileged Execute (RGN_MPL{0-3}.SUX) • Non-secure Privileged Execute (RGN_MPL{0-3}.NSPX) • Non-secure Unprivileged Execute (RGN_MPL{0-3}NSUX) • Address Translation Enable (RGN_TCFG2.ADDR_TRANS_EN)	Firewall Controller
2	Standard region, except for the following fields, of the RWE, are always RAZ/WI: • Secure Privileged Execute (RGN_MPL{0-3}.SPX) • Secure Unprivileged Execute (RGN_MPL{0-3}.SUX) • Non-secure Privileged Execute (RGN_MPL{0-3}.NSPX) • Non-secure Unprivileged Execute (RGN_MPL{0-3}NSUX) • Address Translation Enable (RGN_TCFG2.ADDR_TRANS_EN)	All other Firewall Components

The following table shows the default values of fields, and whether the values can be updated for Regions 0-2 for the Firewall Controller. The tables do not show the RGN_ST register. *Region Status (RGN_ST)* on page Appx-C-326 defines the register. The reset values matching the reset value of the RGN_CTRL{0,1} register.

Table C-61 Region 0 values for Firewall Component 0

Register name	Field name	Region 0	
		Value	Update
RGN_CTRL0	EN	`0b1`	N
RGN_CTRL1	MPE0_EN	`0b1`	N
	MPE1_EN	`0b0`	N
	MPE2_EN	`0b0`	N
	MPE3_EN	`0b0`	N
RGN_LCTRL	LOCK	`0b00`	N
RGN_CFG{0,1}	BASE_ADDR	`0x0`	N
RGN_SIZE	SIZE	UNKNOWN	N
	MULnPO2	`0b1`	N
RGN_TCFG{0,1}	OUTPUT_ADDR/UPPER_ADDR	`0x1_0000` – 2^(FC_CFG1.MNRS+5)	N

Table C-61 Region 0 values for Firewall Component 0 (continued)

Register name	Field name	Region 0	
		Value	Update
RGN_TCFG2	MA_TRANS_EN	0b0	N
	ADDR_TRANS_EN	0b0	N
	INST	0b00	N
	PRIV	0b00	N
	NS	0b00	N
	SH	0b01	N
	MA	0x00	N
RGN_MID0	MST_ID	Configuration MasterID	N
RGN_MPL0	NSUR	0b0	N
	NSUW		N
	NSUX	0b0	N
	NSPR	0b0	N
	NSPW		N
	NSPX	0b0	N
	SUR	0b0	N
	SUW	UNKNOWN	N
	SUX	0b0	N
	SPR	0b1	N
	SPW		N
	SPX	0b0	N
	ANY_MST	0b0	N
RGN_MID{1-3}	MST_ID	UNKNOWN	N

Table C-61 Region 0 values for Firewall Component 0 (continued)

Register name	Field name	Region 0	
		Value	Update
RGN_MPL{1-3}	NSUR	UNKNOWN	N
	NSUW	UNKNOWN	N
	NSUX	0b0	N
	NSPR	UNKNOWN	N
	NSPW	UNKNOWN	N
	NSPX	0b0	N
	SUR	UNKNOWN	N
	SUW	UNKNOWN	N
	SUX	0b0	N
	SPR	UNKNOWN	N
	SPW	UNKNOWN	N
	SPX	0b0	N
	ANY_MST	UNKNOWN	N

Note

RGN_MID{1-3} and RGN_MPL{1-3} are shown here for completeness. Depending on the value of FC_CFG1.NUM_MPE, they may not be implemented and may be Reserved.

Table C-62 Region 1 and 2 values for Firewall Component 0

Register name	Field name	Region 1		Region 2	
		Value	Update	Value	Update
RGN_CTRL0	EN	0b0	Y	0b0	Y
RGN_CTRL1	MPE0_EN	0b0	Y	0b0	Y
	MPE1_EN	0b0	Y	0b0	Y
	MPE2_EN	0b0	Y	0b0	Y
	MPE3_EN	0b0	Y	0b0	Y
RGN_LCTRL	LOCK	0b0	Y	0b0	Y
RGN_CFG{0,1}	BASE_ADDR	0x0	N	0x1_0000	N
RGN_SIZE	SIZE	UNKNOWN	N	UNKNOWN	N
	MULnPO2	0b1	N	0b1	N
RGN_TCFG0	OUTPUT_ADDR/ UPPER_ADDR	0x1_0000 – 2^(FC_CFG1.MNRS+5)	N	0x1_0000 + (NUM_FC * 0x1_0000) – 2^(FC_CFG1.MNRS+5)	N

Table C-62 Region 1 and 2 values for Firewall Component 0 (continued)

Register name	Field name	Region 1		Region 2	
		Value	Update	Value	Update
RGN_TCFG2	MA_TRANS_EN	0b0	N	0b0	N
	ADDR_TRANS_EN	0b0	N	0b0	N
	INST	0b00	N	0b00	N
	PRIV	0b00	N	0b00	N
	NS	0b00	N	0b00	N
	SH	0b01	N	0b01	N
	MA	0x00	N	0x00	N
RGN_MID{0-3}	MST_ID	UNKNOWN	Y	UNKNOWN	Y
RGN_MPL{0-3}	NSUR	UNKNOWN	Y	UNKNOWN	Y
	NSUW	UNKNOWN	Y	UNKNOWN	Y
	NSUX	0b0	N	0b0	N
	NSPR	UNKNOWN	Y	UNKNOWN	Y
	NSPW	UNKNOWN	Y	UNKNOWN	Y
	NSPX	0b0	N	0b0	N
	SUR	UNKNOWN	Y	UNKNOWN	Y
	SUW	UNKNOWN	Y	UNKNOWN	Y
	SUX	0b0	N	0b0	N
	SPR	UNKNOWN	Y	UNKNOWN	Y
	SPW	UNKNOWN	Y	UNKNOWN	Y
	SPX	0b0	N	0b0	N
	ANY_MST	UNKNOWN	Y	UNKNOWN	Y

Note

Only bits log2(MXRS)-1 to log2(MNRS) of the base and upper address are present in the BASE_ADDR, and OUTPUT_ADDR/UPPER_ADDR, fields. All other bits are RAZ.

RGN_MPL{1-3} are shown here for completeness. Depending on the value of FC_CFG1.NUM_MPE, they may not be implemented and may be Reserved.

Fault entries and FWE

C.4.3 Fault entries on page Appx-C-313 and *C.4.7 Registers* on page Appx-C-319 define the fault entries and FWE.

C.11.2 Security Extension

The Security Extension affects the Firewall Controller in the same way as the Firewall Component.

See *C.8 Security Extension* on page Appx-C-365 for more information.

C.11.3 Lockdown Extension

The registers added to the Firewall Controller behave as follows:

- The FC{0-31}_INT_ST registers can be updated in any lockdown state.
- The FC{0-31}_INT_MSK registers are not updateable when in the Partial or Full lockdown state.
- The FW_TMP_CTRL can be updated in any lockdown state.
- The FW_CTRL register is not updated when in the Partial or Full lockdown state.

The Firewall Controller has the same behavior for the Lockdown Extension as *C.9 Lockdown Extension* on page Appx-C-366 describes.

Registers on page Appx-C-379 defines the registers added to the Firewall Controller.

Tamper Interrupt Interface

The Firewall Controller, when LDE.1 or greater is implemented, implements a Tamper Interrupt interface.

The Tamper Interrupt interface indicates when a Tamper or Tamper Overflow interrupt has occurred as indicated by the FW_TMP_CTRL.{TR_VLD,TR_OVERFLW} fields being 1. Software clears the Tamper Interrupt interface by writing 1 to FW_TMP_CTRL.ACK bit.

Tamper report

When LDE.1 or greater is implemented, when a configuration access which meets the following requirements:

- Has passed the protection logic checks of the Firewall Controller.
- Is a supported Configuration Access as defined by the implementation.
- Occurs when FW_SR_CTRL.SR_RDY is 0b1.

Attempts to do one of the following:

- Update an implemented register of a Firewall Component which is in the Partial or Full lockdown state, except for:
  - RWE_CTRL
  - FE_CTRL
  - EDR_CTRL
  - FC{0-31}_INT_ST
  - FW_TMP_CTRL
  - LD_CTRL, when Lockdown interface is not asserted.

  For more information see *C.9.1 Firewall Component lockdown* on page Appx-C-366.
- Update an implemented register of a region which is in the locked state. For more information see *C.9.2 Region lockdown* on page Appx-C-367.

  Note

  This includes if the register is a read-only register.

- Update the value of the RGN_LCTRL.LOCK field when it is set to 1 and the Firewall Component is in the Partial lockdown state.
- Update the value of the RGN_LCTRL.LOCK field when the Firewall Component is in the Full lockdown state.
- Change the lockdown state of a Firewall Component, when it is Partial or Full, when LD_CTRL.LDI_ST is 1 in the Firewall Controller.
- Access the Tamper Report registers with an access without the correct properties or from a master with a MasterID other than in region 0.

When one of these events occurs the Firewall Controller:

- Generates a Configuration Access Error to configuration access.
- Generates a Tamper interrupt and Tamper report, if there is not a valid Tamper report present. Otherwise a Tamper Overflow interrupt is generated.

A Tamper report is valid when the FW_TMP_CTRL.TR_VLD is set to 1. Software is required to acknowledge the Tamper report, only when it has collected the information contained in the FW_TMP_{TA,TP,MID,CTRL} registers. Once software has acknowledged the Tamper report, the values in the FW_TMP-{TA,TP,MID,CTRL} registers become 0. Except if another Tamper report is generated in the same cycle.

Registers

The table below summaries the registers added to the Firewall Control for the Tamper Report.

Table C-63 Tamper Report register summary

Offset	Short name	Access	Name
0xE90	*FW_TMP_TA* on page Appx-C-379	RO	Firewall Tamper Transaction Address
0xE94	-	RO	Reserved
0xE98	*FW_TMP_TP* on page Appx-C-380	RO	Firewall Tamper Transaction Properties
0xE9C	*FW_TMP_MID* on page Appx-C-380	RO	Firewall Tamper MasterID
0xEA0	*FW_TMP_CTRL* on page Appx-C-380	RW	Firewall Tamper Control

These registers are only accessible by transactions with the following properties:

- MasterID as defined in region 0
- When SE.1 or greater is implemented, with a security property set to secure
- Privileged

Any access to these registers which does not meet these requirements:

- Generates a Configuration Access Error
- Generates a tamper report:
  - If there is not already a valid tamper report
  - If there is a valid tamper report, which software is acknowledging this cycle
- Generates a tamper report overflow, if there is already a valid tamper report, which software is not acknowledging this cycle.

These registers are only implemented when LDE.1 or greater is implemented, otherwise they are Reserved and generate a Configuration Access Error if accessed.

Firewall Tamper Transaction Address (FW_TMP_TA)

The following table gives a bit-level description of the Firewall Tamper Transaction Address (FW_TMP_TA) register.

Table C-64 Firewall Tamper Transaction Address register

Bits	Name	Description	Type	Reset
[31:21]	-	Reserved	RO	0x00
[20:2]	TMP_TRANS_ADDR	Address of the accessed register which caused the tamper report to be generated	RO	UNKNOWN
[1:0]	-	Reserved	RO	00

Firewall Tamper Transaction Properties (FW_TMP_TP)

The following table gives a bit-level description of the Firewall Tamper Transaction Properties (FW_TMP_TP) register.

Table C-65 Firewall Tamper Transaction Properties register

Bits	Name	Description	Type	Reset
[31:20]	-	Reserved	RO	0x000
[19:18]	-	Reserved	RO	0x00
[17]	PRIV	Indicates the privileged level of the transaction which generated the tamper report. 0b0: Unprivileged 0b1: Privileged	RO	UNKNOWN
[16]	NS	Indicates the security level of the transaction which generated the tamper report. 0b0: Secure 0b1: Non-secure	RO	UNKNOWN
[15:0]	-	Reserved	RO	0x0000

Firewall Tamper MasterID (FW_TMP_MID)

The following table gives a bit-level description of the Firewall Tamper MasterID (FW_TMP_MID) register.

Table C-66 FW_TMP_MID register

Bits	Name	Description	Type	Reset
[31:0]	MST_ID	Indicates the MasterID of the transaction which caused the tamper report.The width of this field depends on the value of FC_CFG2. MST_ID_WIDTH. Any unused bits are Reserved and treated as RAZ/WI.	RO	The reset value of this field depends on the value of FC_CFG2. SINGLE_MST: 0b0: UNKNOWN 0b1: Fixed valued defined at design time.

Firewall Tamper Control (FW_TMP_CTRL)

The following table gives a bit-level description of the Firewall Tamper Control (FW_TMP_CTRL).

Table C-67 FW_TMP_CTRL register

Bits	Name	Description	Type	Reset
[31]	TR_VLD	Indicates whether there is a valid tamper report or not. `0b0`: No valid tamper report `0b1`: Valid tamper report When this field is 0 the values in the following registers read as 0: • FW_TMP_TA • FW_TMP_TP • FW_TMP_MID	RO	0
[30]	TR_OVERFLW	Indicates whether a tamper report overflow has occurred `0b0`: No tamper report overflow has occurred `0b1`: Tamper report overflow has occurred	RO	0
[29:1]	-	Reserved	RO	`0x0000_0000`
[0]	ACK	Acknowledge the Tamper report This field always reads as 0. Writes to this field behave as follows: `0b0`: Ignored `0b1`: Tamper report acknowledged Writes to this register are ignored if FW_TMP_CTRL.TR_VLD is 0.	WO	0

C.11.4 Translation Extension

A Firewall Controller can support any level of the Translation Extension. However, no translation can be enabled for Regions 0 to 2. For these regions, the RGN_TCFG2 register is read-only.

Translation Extension support is the same as for a Firewall Component. See *C.6 Translation Extension* on page Appx-C-356.

C.11.5 Region Size Extension

A Firewall Controller always supports RSE.1.

For more information on RSE see *C.7 Region Size Extension* on page Appx-C-364.

C.11.6 Interrupts

The Firewall Controller receives Interrupt requests from the Firewall Components, on the Firewall Configuration interfaces.

When the Firewall Controller receives an Interrupt requests, and the interrupt is not masked, it must:
- Generate an interrupt, on the Interrupt interface
- Update the interrupt status registers

Changing the value of the interrupt mask does not affect any interrupts which have already been asserted.

All interrupts in the Firewall are considered level sensitive, and require software to acknowledge the interrupt by writing a value of 1 to the corresponding interrupt status field. If the Firewall Controller receives a request from a Firewall Component to generate an interrupt at the same time as software writes 1 to the status bit the interrupt must remain asserted and the interrupt status field is set to 1.

This includes the case where software acknowledges the interrupt in the same cycle as the request is received.

Generation

Interrupts are generated by the Firewall Components and reported to the Firewall Controller.

A Firewall Component can generate the following types of interrupts:

- When PE.1 or greater is implemented:
  - Access Error
  - Programming Error
  - Fault Entry Overflow
- When ME.1 or greater is implemented:
  - Error Detection
  - Error Detection Overflow
- Tamper, when LDE.1 or greater is implemented.
- Tamper Overflow, when LDE.1 or greater is implemented.

Access Error

The Access Error interrupt is generated when:

- A transaction enters the Faulted state and the Firewall Component is configured to generate a fault entry
- If there is at least one invalid fault entry in the Firewall Component

If all fault entries are valid, then a Fault Entry Overflow interrupt is generated instead, see *Fault Entry Overflow* on page Appx-C-382.

Programming Error

The Programming Error interrupt is generated when a transaction matches more than one region and MPE pair.

The Programming Error is also generated:

- When the Firewall Component is configured to generate a fault entry
- If there is at least one invalid fault entry in the Firewall Component

If all fault entries are valid, then a Fault Entry Overflow interrupt is generated instead, see *Programming Error* on page Appx-C-382.

Fault Entry Overflow

The Fault Entry Overflow interrupt is generated when the Firewall Component attempts to generate a fault entry, due to:

- The Firewall Component having the conditions to generate an Access Error interrupt, except for all fault entries being valid.
- The Firewall Component having the conditions to generate a Programming Error interrupt, except for all fault entries being valid.

Error Detection

The Error Detection interrupt is generated when the Firewall Component detects an error response to a transaction, the monitoring logic is enabled, and there is at least one invalid error detection report.

If all error detection reports are valid, then an Error Detection Overflow interrupt is generated instead, see *Error Detection* on page Appx-C-382.

Error Detection Overflow

The Error Detection Overflow interrupt is generated when the Firewall Component detects an error response to a transaction, the monitoring logic is enabled, and all error detection reports are valid.

Tamper

The Tamper interrupt is generated when a configuration access attempts to do any of the following and FW_TMP_CTRL.TR_VLD bit is 0:

- Update any register of a Firewall Component when LDE.1 or greater is implemented and the Firewall Component is in the Partial or Full lockdown state, except for the following registers:
  - RWE_CTRL
  - FE_CTRL
  - EDR_CTRL
  - FC{0-31}_INT_ST
  - FW_TMP_CTRL
  - LD_CTRL, when Lockdown interface is not asserted
- Update a field of a region which is in the Locked state, when LDE.2 and PE.1 or greater, are implemented.
- Update the value of the RGN_LCTRL.LOCK field when it is set to 1 and the Firewall Component is in the Partial lockdown state.
- Update the value of the RGN_LCTRL.LOCK field when the Firewall Component is in the Full lockdown state.
- Change the lockdown state of a Firewall Component, when it is Partial or Full, when LD_CTRL.LDI_ST is 1 in the Firewall Controller.
- Access the Tamper Report registers with an access without the correct properties or from a master with a MasterID other than in region 0.

If FW_TMP_CTRL.TR_VLD bit is 1, then a Tamper Overflow interrupt is generated instead, see section *Tamper* on page Appx-C-383.

The Tamper interrupt is only supported when LDE.1 or greater is implemented.

Tamper Overflow

The Tamper Overflow interrupt is generated when a configuration access attempts to perform an event listed in *Tamper* on page Appx-C-383 and the FW_TMP_CTRL.TR_VLD field is 1.

C.11.7 Registers

This section describes the extra registers defined for Firewall Controller compared with a Firewall Component.

A Firewall Controller includes all the registers present in a Firewall Component. In this section the additional registers are covered.

The Firewall Controller implements registers for:

- Firewall Control and Status of the whole Firewall
- Interrupts
- Identification

The Control and Status registers allow software to configure:

- Behavior of the Firewall Controller, when a Configuration Access Error is generated by itself or another Firewall Component.
- When SRE.1 is implemented, whether the shadow registers are required to retain their values when the Firewall Controller enters the Disconnected state.

The Interrupt registers allow software to configure and receive interrupts generated by the Firewall Components in the Firewall.

The Tamper register, logs information about tamper transactions.

The identification registers provide the following information:

- Architecture version the implementation of the Firewall is compliant to
- Details about the implementor of the Firewall
- Identification registers

The table below summarizes the registers added to the Firewall Controller over a standard Firewall Component.

Note

This excludes the Tamper Report registers which were defined in *Registers* on page Appx-C-379.

Table C-68 Summary of the Firewall Controller registers

Offset	Short name	Access	Name
Control and Status			
0x000	*FW_CTRL* on page Appx-C-384	RW	Firewall Control
0x004	*FW_ST* on page Appx-C-385	RO	Firewall Status
0x00C	*FW_SR_CTRL* on page Appx-C-385	RW	Firewall Shadow Register Control
Interrupts			
0xD00- 0xD7C	*FC{0-31}_INT_ST* on page Appx-C-386	RW	Firewall Component 0-31 Interrupt Status
0xD90	*FC_INT_ST* on page Appx-C-387	RO	Firewall Interrupt Status
0xE00- 0xE7C	*FC{0-31}_INT_MSK* on page Appx-C-387	RW	Firewall Component 0 -31 Interrupt Mask
Identification			
0xFC8	*IIDR* on page Appx-C-388	RO	Implementation Identification Register
0xFCC	*AIDR* on page Appx-C-389	RO	Architecture Identification Register
0xFD0	*PID4* on page Appx-C-389	RO	Peripheral ID4
0xFD4	*PID5* on page Appx-C-389	RO	Peripheral ID5
0xFD8	*PID6* on page Appx-C-389	RO	Peripheral ID6
0xFDC	*PID7* on page Appx-C-390	RO	Peripheral ID7
0xFE0	*PID0* on page Appx-C-390	RO	Peripheral ID0
0xFE4	*PID1* on page Appx-C-390	RO	Peripheral ID1
0xFE8	*PID2* on page Appx-C-390	RO	Peripheral ID2
0xFEC	*PID3* on page Appx-C-390	RO	Peripheral ID3
0xFF0	*CID0* on page Appx-C-391	RO	Component ID0
0xFF4	*CID1* on page Appx-C-391	RO	Component ID1
0xFF8	*CID2* on page Appx-C-391	RO	Component ID2
0xFFC	*CID3* on page Appx-C-391	RO	Component ID3

Firewall Control (FW_CTRL) register

The following table gives a bit-level description of the Firewall Control (FW_CTRL) register.

Table C-69 FW_CTRL register

Bits	Name	Description	Type	Reset
[31:2]	-	Reserved	RO	0x0000_0000
[1]	RAZ	Configures the value returned for read accesses which generate a Configuration Access Error. 0b0: Read data is based on the StreamID 0b1: Read data is all 0s	RW	0
[0]	ERR	Configures the response for transactions which generate a Configuration Access Error. 0b0: No error 0b1: Error	RW	1

Firewall Status (FW_ST) register

The following table gives a bit-level description of the Firewall Status (FW_ST) register.

Table C-70 FW_ST register

Bits	Name	Description	Type	Reset
[31:2]	-	Reserved	RO	0x0000_0000
[1]	RAZ	Indicates the value returned for read accesses which generate a Configuration Access Error. 0b0: Read data is based on the StreamID 0b1: Read data is all 0s	RO	0
[0]	ERR	Indicates the response for a transaction which generate a Configuration Access Error. 0b0: No error 0b1: Error	RO	1

Note

When the Firewall Controller implements ME.1, or greater, and FW_ST.ERR is 1, any access which generates a Configuration Access Error, will also generate an error detection report and associated Error Detection interrupt, if the monitor logic is enabled.

Firewall Shadow Register Control (FW_SR_CTRL) register

The following table gives a bit-level description of the Firewall Shadow Register Control (FW_SR_CTRL) register.

This register is only implemented when SRE.1 is implemented. Otherwise, it is Reserved and generates a Configuration Access Error when accessed.

Table C-71 FW_SR_CTRL register

Bits	Name	Description	Type	Reset
[31]	SR_RDY	Indicates when the shadow registers are ready. `0b0`: Shadow registers are not ready. `0b1`: Shadow registers are ready. For more information on this field see *C.10.4 Shadow Register Initialization* on page Appx-C-371.This field is Reserved and treated as RAZ/WI when SRE.0 is implemented.	RO	0
[30:1]	-	Reserved	RO	`0x0000_0000`
[0]	SR_PWR	Shadow register power request when Firewall Controller enters the Disconnected state: `0b0`: No request for the shadow registers to retain their values the next time the Firewall Controller enters the Disconnected state. `0b1`: Request for the shadow registers to retain their values the next time the Firewall Controller enters the Disconnected state. This field is Reserved and treated as RAZ/WI when SRE.0 is implemented.	RW	0

Firewall Component {0-31} Interrupt Status (FC{0-31}_INT_ST) register

The following table gives a bit-level description of the Firewall Component {0-31} Interrupt Status (FC{0-31}_INT_ST) register.

There is an interrupt status register per Firewall Component implemented in the Firewall. The fields within this register indicate the status of the interrupts generated by that Firewall Component. The register is only implemented when the associated Firewall Component is implemented, otherwise this register is Reserved and generates a Configuration Access Error when accessed.

Note

Software can find out how many Firewall Components are implemented in the Firewall by reading the FC_CFG3.NUM_FC field of the Firewall Controller.

Table C-72 FC{0-31}_INT_ST register

Bits	Name	Description	Type	Reset
[31:5]	-	Reserved	RO	`0x000_0000`
[4]	ED_OVRFLW_ST	Indicates whether an error detection report overflow has occurred in the Firewall Component. `0b0`: No error detection report overflow has occurred `0b1`: Error detection report overflow has occurred This field is write 1 to clear, writing 0 to this field has no effect.	RW	0
[3]	ED_ST	Indicates whether the Firewall Component has detected an error response and generated an error detection report. `0b0`: No error response has been detected `0b1`: Error response has been detected This field is write 1 to clear, writing 0 to this field has no effect.	RW	0

Table C-72 FC{0-31}_INT_ST register (continued)

Bits	Name	Description	Type	Reset
[2]	FLT_OVRFLW_ST	Indicates whether a fault entry overflow has occurred in the Firewall Component. `0b0`: No fault entry overflow has occurred `0b1`: Fault entry overflow has occurred This field is write 1 to clear, writing 0 to this field has no effect.	RW	0
[1]	PROG_ERR_ST	Indicates whether the Firewall Component has detected a Programming Error. `0b0`: No Programming Error has occurred. `0b1`: Programming Error has occurred. This field is write 1 to clear, writing 0 to this field has no effect.	RW	0
[0]	ACC_ERR_ST	Indicates whether the Firewall Component has detected an Access Error. `0b0`: No Access Error has occurred. `0b1`: An Access Error has occurred. This field is write 1 to clear, writing 0 to this field has no effect.	RW	0

For all the fields in the FC{0-31}_INT_ST, if software attempts to clear the interrupt at the same time as the Firewall Controller receives a new interrupt request, the field remains set.

Firewall Interrupt Status (FW_INT_ST) register

The following table gives a bit-level description of the Firewall Interrupt Status (FW_INT_ST) register.

The fields within this register indicate the status of the interrupts generated by the connected Firewall Component. When any bit, in the associated FC{0-31}_INT_ST register is asserted, the corresponding bit in the FW_INT_ST register will be asserted.

Table C-73 FW_INT_ST register

Bits	Name	Description	Type	Reset
[31:0]	FC_INT_ST{x}	Interrupt status for the associated Firewall Component. There is a bit per Firewall Component which indicates if any bit in the associated FC{0-31}_INT_ST register is 1. Any bits associated with a Firewall Component which is not implemented are Reserved and treated as RAZ/WI. `0b0`: No interrupt is asserted by Firewall Component. `0b1`: Interrupt is asserted by Firewall Component. Bit 0 is for Firewall Component 0 whilst bit 31 is for Firewall Component 31.	RO	`0x0000_0000`

Note

Software can find out how many Firewall Components are implemented in the Firewall by reading the FC_CFG3.NUM_FC field of the Firewall Controller.

Firewall Component {0-31} Interrupt Mask (FC{0-31}_INT_MSK) register

The following table gives a bit-level description of the Firewall Component {0-31} Interrupt Mask (FC{0-31}_INT_MSK) register.

There is an interrupt mask register per Firewall Component implemented in the Firewall. The fields within this register, control whether an interrupt is generated (for that type of interrupt) for the respective Firewall Component. The register is only implemented when the associated Firewall Component is implemented, otherwise this register is Reserved and generates a Configuration Access Error when accessed.

Note

Software can find out how many Firewall Components are implemented in the Firewall by reading the FC_CFG3.NUM_FC field of the Firewall Controller.

Table C-74 FC{0-31}_INT_MSK register

Bits	Name	Description	Type	Reset
[31:5]	-	Reserved	RO	0x000_000
[4]	ED_OVRFLW_MSK	Selects whether Error Detection Overflow interrupts are masked. 0 – Error Detection Overflow interrupts are reported to the system. 1 – Error Detection Overflow interrupt is not reported to the system. This field is Reserved and treated as RAZ/WI when ME.0 is implemented.	RW	0
[3]	ED_MSK	Selects whether Error Detection interrupts are masked. 0 – Error Detection interrupts are reported to the system. 1 – Error Detection interrupt is not report to the system. This field is Reserved and treated as RAZ/WI when ME.0 is implemented.	RW	0
[2]	FLT_OVRFLW_MSK	Selects whether Fault Event Overflow interrupts are masked. 0 – Fault Event Overflow interrupts are reported to the system. 1 – Fault Event Overflow interrupts are not reported to the system. This field is Reserved and treated as RAZ/WI when PE.0 is implemented.	RW	0
[1]	PROG_ERR_MSK	Selects whether Programming Error interrupts are masked. 0 – Programming Error interrupts are reported to the system. 1 – Programming Error interrupts are not reported to the system. This field is Reserved and treated as RAZ/WI when PE.0 is implemented.	RW	0
[0]	ACC_ERR_MSK	Selects whether Access Error interrupts are masked. 0 – Access Error interrupts are reported to the system. 1 – Access Error interrupts are not reported to the system. This field is Reserved and treated as RAZ/WI when PE.0 is implemented.	RW	0

When an interrupt is masked the associated bit in the FC{0-31}_INT_ST register is also not set. However, if the associated bit in the FC{0-31}_INT_ST is already 1, setting the mask bit to 1 does not change the value of the bit in the FC{0-31}_INT_ST register.

Implementation Identification (IIDR) register

The following table gives a bit-level description of the Implementation Identification (IIDR) register.

Table C-75 IIDR register

Bits	Name	Description	Type	Reset
[31:20]	PRODUCT_ID	Firewall part ID	RO	0x075
[19:16]	VARIANT	Firewall varient	RO	0x0
[15:12]	REVISION	Firewall revision	RO	0x1
[11:0]	IMPLEMENTER	Contains the JEP106 code of Arm. • [11:8] JEP106 continuation code of implementer • [7] Always 0 • [6:0] JEP106 identity code of implementer	RO	0x43B

Architecture Identification (AIDR) register

The following table gives a bit-level description of the Architecture Identification (AIDR) register.

Table C-76 AIDR register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	ARCH_MAJOR_REV	Firewall Architecture Major Revision 0x0 – Major Revision 0 All other values are reserved.	RO	0x0
[3:0]	ARCH_MINOR_REV	Firewall Architecture Minor Revision 0x0 – Minor Revision 0 All other values are reserved.	RO	0x0

Peripheral ID 4 (PID4) register

The following table gives a bit-level description of the Peripheral ID 4 (PID4) register.

Table C-77 PID4 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	Size	Number of 4KB occupied by the System ID block. This field is deprecated.	RO	0x0
[3:0]	DES_2	JEP Continuation. For Arm this field is 0x4.	RO	0x4

Peripheral ID 5 (PID5) register

The following table gives a bit-level description of the Peripheral ID 5 (PID5) register.

Table C-78 PID5 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 6 (PID6) register

The following table gives a bit-level description of the Peripheral ID 6 (PID6) register.

Table C-79 PID6 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 7 (PID7) register

The following table gives a bit-level description of the Peripheral ID 7 (PID7) register.

Table C-80 PID7 register

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RO	0x0000_0000

Peripheral ID 0 (PID0) register

The following table gives a bit-level description of the Peripheral ID 0 (PID0) register.

Table C-81 PID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PART_0	Bits [7:0] of part ID	RO	0x75

Peripheral ID 1 (PID1) register

The following table gives a bit-level description of the Peripheral ID 1 (PID1) register.

Table C-82 PID1 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	DES_0	Bits [3:0] of JEP 106 Identity	RO	0xB
[3:0]	PART_1	Bits [11:8] of part ID	RO	0x0

Peripheral ID 2 (PID2) register

The following table gives a bit-level description of the Peripheral ID 2 register.

Table C-83 PID2 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	REVISION	Major revision of the System ID block.	RO	0x0
[3]	JEDEC	Indicates the use of JEDEC JEP106 identification scheme.	RO	0b1
[2:0]	DES_1	Bits [6:4] of JEP 106 Identity. For Arm this value is 0b011.	RO	0b011

Peripheral ID 3 (PID3) register

The following table gives a bit-level description of the Peripheral ID 3 (PID3) register.

Table C-84 PID3 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	REVAND	Minor revision of the System ID block	RO	0x1
[3:0]	CMOD	Customer modification field	RO	0x0

Component ID 0 (CID0) register

The following table gives a bit-level description of the Component ID 0 (CID0) register.

Table C-85 CID0 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_0	Preamble 0	RO	0x0D

Component ID 1 (CID1) register

The following table gives a bit-level description of the Component ID 1 (CID1) register.

Table C-86 CID1 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:4]	CLASS	Class of the component.	RO	0xF
[3:0]	PRMBL_1	Preamble 1	RO	0x0

Component ID 2 (CID2) register

The following table gives a bit-level description of the Component ID 2 (CID2) register.

Table C-87 CID2 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_2	Preamble 2	RO	0x05

Component ID 3 (CID3) register

The following table gives a bit-level description of the Component ID 3 (CID3) register.

Table C-88 CID3 register

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	RO	0x00_0000
[7:0]	PRMBL_3	Preamble 3	RO	0xB1

C.11.8 Changing Configurations Settings of Firewall

Changing various Configurations Settings of the Firewall

Changing Configuration Access Error Response Type

Requirements of changing Configuration Access Errors response type.

The Firewall Components generates Configuration Access Errors when illegal transactions are generated. When a Configuration Access Error is generated by a transaction, the Firewall Controller can be configured to generate an error response or not, when responding to the transaction. Software uses the FW_CTRL.ERR field to configure this behavior. The FW_ST.ERR field indicates the current response type the Firewall Controller uses to respond to Configuration Access Errors generated by the Firewall Components. When FW_CTRL.ERR and FW_ST.ERR are different, the Firewall is changing its Configuration Access Error response behavior.

The value of FW_ST.ERR only updates to the value of FW_CTRL.ERR, when any new Configuration Access Errors use the new value configured in the FW_CTRL.ERR field. Any transaction which has already generated a Configuration Access, behaves as defined by the value of FW_ST.ERR at the point the Firewall Controller is notified about the Configuration Access Error.

If software attempts to change the value of FW_CTRL.ERR back to its previous value, when it differs from FW_ST.ERR, the Firewall treats the update as a Configuration Access Error and does not update the FW_CTRL register.

Arm strongly recommends:

- Once software changes the value of either FW_CTRL.ERR, it waits for the value of FW_ST.ERR to reflect this change, before changing the value again.
- When software changes the values of FW_CTRL.ERR there are no outstanding transactions to the Firewall Controller, other than the configuration access performing the update. The method software uses to achieve this is system-dependent. If there are other outstanding transactions it is UNPREDICTABLE whether any Configuration Access Errors use the new or old value of the FW_ST.ERR field.

Changing Configuration Access Error Read Data Response

Requirements of changing Configuration Access Errors response type.

When a read transaction, to the registers of the Firewall, generates a Configuration Access Error, it is configurable whether the read data is set to all 0s or a value dependent on the StreamID of the transaction. Software uses the FW_CTRL.RAZ field to configure this behavior. The FW_ST.RAZ field indicates what the read data for a read transaction, which generates a Configuration Access Error, is set to. When FW_CTRL.RAZ and FW_ST.RAZ are different, the Firewall Controller is changing its Configuration Access Error read response behavior.

The value of FW_ST.RAZ only updates, to the value of FW_CTRL.RAZ, when any new read transactions which become a Configuration Access Error, will be treated as the value configured in the FW_CTRL.RAZ field. Any transaction which has already generated a Configuration Access, behaves as defined by the value of FW_ST.RAZ at the point the Firewall Controller is notified about the Configuration Access Error.

If software attempts to change the value of FW_CTRL.RAZ, back to its previous value, when it differs from FW_ST.RAZ, the Firewall treats the update as a Configuration Access Error and does not update the FW_CTRL register.

Arm strongly recommends:

- Once software changes the value of either FW_CTRL.RAZ, it waits for the value of PE_ST.RAZ to reflect this change, before changing the value again.
- When software changes the values of PE_CTRL.RAZ there are no outstanding transactions to the Firewall Controller, other than the configuration access performing the update. The method software uses to achieve this is system-dependent. If there are other outstanding transactions it is UNPREDICTABLE whether any Configuration Access Errors use the new or old value of the FW_ST.RAZ field.

Changing Shadow Register Power Behavior

Requirements of changing shadow register power behavior.

When SRE.1 is implemented, using the FW_SR_CTRL.SR_PWR field enables software to configure whether the Firewall Controller accepts entering a power mode where the shadow registers may lose their contents.

C.12 Software usage

This section covers the software usage model for the Firewall.

Note

Arm strongly advises that if the Firewall programming code is updateable that software perform checks to validate the code before executing the code. Failure to do so, can reduce the effectiveness of the Firewall.

This section contains the following subsections:

C.12.1 Fault usage model

When a transaction is terminated by the Firewall Component (PE_ST.FLT_CFG is `0b10`), a fault entry may be generated alongside an Access or Programming Error interrupt.

Software should read the fault entry and log the details of the transaction which faulted. Once software has logged the details, software is expected to acknowledge the fault entry. When the fault entry is acknowledged, the fault entry becomes invalid and is available to be used for another faulting transaction.

C.12.2 Error detection report usage

Error detection reports are intended to allow software to discover when a master in the SoC, has performed an operation which a slave in the SoC has generated an error to.

An example, of a slave in a SoC which generates errors to transactions is a Default Slave. A Default Slave is a peripheral, which transactions are routed to when the address of the transaction does not map to any other slave in the SoC. Typically, a Default Slave generates an error to complete the transaction and avoid a bus lockup. For a read transaction this includes generating the read data which is typically all 0s. For many processors, a value of all 0s is treated as a NOP operation if the data is executed. This means that it is possible for malicious software agents to use slaves, like the Default Slave, to perform a NOP slide attack and gain access to areas or privileges that they would not normally be allowed.

Using the error detection reports generated by the Firewall, a software agent can detect when this is occurring and investigate and prevent the NOP slide attack from gaining the access or privilege.

It is expected, that when software receives an Error Detection interrupt, it reads the contents of the error detection report and either:

- Logs the information for analysis.
- Takes an action, such as disabling the master.
- Performs both

Note

Software can be implemented to log the information by default. However, after a certain number of error transactions from the same master in a certain amount of time, it disables the master and investigates further.

Independent of what action software takes, Arm strongly recommends that software acknowledges the error detection report in a timely manner to reduce the chance of getting an Error Detection Overflow interrupt.

C.12.3 Region programming

When programming a region, software must follow these steps:

1. Select the correct region using the RWE_CTRL register.
2. Program the region base address using the RGN_CFG{0,1} registers.
3. Program the region size or upper address using the RGN_SIZE or RGN_TCFG{0,1} registers.

   —————— **Note** ——————

   For regions in a Firewall Component implementing PE.1, setting the base address and size are not required, as the region base address, size and upper address are pre-defined.

   ————————————————

4. Program and enable the translation properties, if required, using the RGN_TCFG2 register. If the region is required to perform address translation software must use the RGN_TCFG{0,1} to define the address translation.
5. Program the required Master Permission Entries using the RGN_MID{0-3} and RGN_MPL{0-3} registers.
6. Enable the required Master Permission Entries using the RGN_CTRL1.MPE_EN{0-3} bits.
7. Wait for the associated RGN_ST.MPE_EN{0-3} bits to become 1.
8. Enable the region, by setting the RGN_CTRL0.EN to 1.
9. Wait for RGN_ST.EN to become 1.

If software is required to update a region or MPEs settings, a break-before-make method is used.

—————— **Note** ——————

The break-before-make method means that software sets certain enables to 0 before updating the values. This prevents a transaction being checked against the regions or MPEs previous values.

————————————————

Depending on the settings of the region which are required to be updated, the sequence to follow for the break before-make-method is different.

Update Master Permission Entry

Required software sequence of updating a *Master Permission Entry* MPE.

When software is only updating an MPE the sequence is as follows:

1. Select the correct region using the RWE_CTRL register.
2. Disable the MPE, by setting the associated RGN_CTRL1.MPE_EN{0-3} bit to 0. Software may need to unlock the region first.
3. Wait for the associated RGN_ST.MPE_EN{0-3} bit to become 0.
4. Update the values in the RGN_MID{0-3} and RGN_MPL{0-3} register.
5. Enable the MPE, by setting the associated RGN_CTRL1.MPE_EN{0-3} bit to 1.
6. Wait for the associated RGN_ST.MPE_EN{0-3} bit to become 1.

Arm recommends that before software starts the above sequence. It requests that the master or masters associated with the MasterID in the RGN_MID{0-3} to be updated, stop issuing new transactions and wait for any outstanding transactions to complete. This removes the possibility for a transaction being checked against an incorrect or missing MPE programming.

Update region address range or translation

When software is required to update any region setting, other than an MPE, the sequence is as follows:

1. Select the correct region using the RWE_CTRL register.
2. Disable the region, by setting the RGN_CTRL0.EN bit to 0. Software may need to unlock the region first.
3. Wait for the RGN_ST.EN bit to become 0.
4. Update the required settings in the RGN_SIZE, RGN_CFG{0,1} and RGN_TCFG{0-2} registers, as required.

5. Enable the region, by setting the RGN_CTRL0.EN bit to 1. At this point software can also lock, or re-lock the region.
6. Wait for the RGN_ST.EN bit to become 1.

Arm recommends that before software starts the above sequence, it requests that all masters which can generate accesses that are checked against the region, stop issuing new transactions and wait for any outstanding transaction to complete. This removes the possibility for a transaction being checked against the incorrect or missing region programming.

Master and region restrictions

This section describes software considerations when multiple masters are implemented and multiple regions are defined.

Depending on the configuration and integration of the Firewall in a system, software may be required to:

- Program the regions of the Firewall so that no transaction issued by a master ever matches in more than one region.
- Prevent a master issuing a legal transaction which is not wholly contained within a single region of Firewall Components it passes through.

For example, there are three masters in a system A, B and C. Master A communicates with master B and C using separate 1KB regions which are contiguous in memory. Master A can generate any size transactions as long as it does not cross a 2KB boundary. Masters B and C can only generate any size transaction as long as it does not cross 1KB boundary. Software must either:

- Define 2 regions:
  - 1 region allowing master A and B access to a 1KB region.
  - 1 region allowing master A and C access to a 1KB region.

  Also prevent master A from ever issuing a transaction which crosses the 1KB boundary.
- Define 3 regions:
  - 1 region allowing master A access to a 2KB region.
  - 1 region allowing master B access to the lower 1KB of that region.
  - 1 region allowing master C access to the upper 1KB of master A region.

C.12.4 Bypass

A Firewall Component which supports PE.1 or greater also supports a mechanism to bypass the protection logic checks, for the purpose of SoC debug.

——— **Note** ———

Arm strongly recommends that software uses the bypasses feature carefully as it disables all protection provided by the Firewall and can lead to exposing information to agents who would not normally have access.

———————————

Bypass mask

The PE_ST.BYPASS_MSK field is intended to mask the Bypass interface value of the Firewall Component.

Software can use this field to:

- Prevent the Firewall Component from becoming bypassed. This is typically the case in a production device where software sets the field to 1 during initial configuration of the Firewall Component. Arm strongly recommends that software only sets PE_ST.BYPASS_MSK during initial configuration, if it can guarantee that there will never be a requirement to perform SoC debug without assistant software that is able to re-configure the Firewall.
- During debug, return a Firewall Component which is bypassed, to not being bypassed. This is typically the case during SoC debug, where the debug agent wants to re-enable functionality of the

Firewall. This could be to enable translation of transactions or to test the configuration of the Firewall programming.

PE_BPS usage

The PE_BPS register provides an indication of whether the Firewall Component is bypassed or not and the current value of the Bypass interface.

The value read from this register must be considered as a snapshot in time; software cannot rely on the value as the only indication of whether a Firewall Component is bypassed or not. The value of PE_BPS must be used in the context of the wider system.

If the Firewall supports SRE.1 then the value read when a Firewall Component is not in the Connected state could indicate the Firewall Component is not bypassed, but once the Firewall Component returns to the Connected state, it could become bypassed. Software must ignore the value of PE_BPS register if PE_BPS.BYPASS_VLD is 0.

Arm strongly recommends the following:

- When bypassing a Firewall Component the following sequence is followed:
  - Set PE_CTRL.BYPASS_MSK to 0, if not already done.
  - Update the value of the Bypass interface. The method by which this is achieved is system-dependent and outside the scope of this document.
  - Check that PE_BPS.BYPASS_ST reads as 1.
- When returning a Firewall Component to not-be-bypassed, one of the following sequences is performed:
  - Updated the value of the Bypass interface and wait for PE_BPS.BYPASS_ST to read as 0.
  - Set PE_CTRL.BYPASS_MSK to 1 and wait for PE_BPS.BYPASS_ST to read as 0.

  Which method is used depends on the reason why the Firewall Component is returning to not-to-be bypassed and how the Firewall is integrated into the system.

C.12.5 Unknown values

Some fields in the Firewall have UNKNOWN reset values.

Software must never rely on the UNKNOWN reset value and must always set fields to a known value before performing other activities. Failure to do so can lead to UNPREDICTABLE results. Software must obey the following rules:

- Before enabling an MPE, it must:
  - Set the RGN_MPL fields with an UNKNOWN reset value to a known value.
  - Set either:
    - RGN_MID{0-3} to a known value.
    - RGN_MPL.ANY_MST to 1.
- Before enabling a region, it must:
  - Set the RGN_CFG{0,1} to a known value.
  - Set the RGN_TCFG{0,1} to a known value, if:
    - RGN_TCFG2.ADDR_TRANS_EN is set to `0b1`.
    - RGN_SIZE.MULnPO2 is set to `0b1`.

Note

When the Firewall implements SRE.1, fields which are part of the shadow registers can have a different value between the copy of the field in the shadow registers and the Firewall Component until the register is set to a value by software.

Read-only fields which have an UNKNOWN are not used by the hardware and software must not rely on the value in these fields.

C.13 Programmers model overview

This section summarizes the programmers model.

The programmers model for the Firewall is dependent upon the number and type of Firewall Components which are part of the Firewall. The Firewall occupies 2MB of address space in total. Each Firewall Component is allocated 64KB page, but only uses the first 4KB, with the remaining 60KB being Reserved. The Firewall Components are arranged by the Firewall Component ID, in ascending order. With the Firewall Controller allocated to the first 64KB page. The Firewall always occupies 2MB with any unused 64KB pages being Reserved.

Any memory marked as Reserved will generate a Configuration Access Error if accessed. This includes the following conditions:

- Locations which are part of the Firewall memory map but are not allocated to an implemented Firewall Component.
- Locations which are always Reserved, as defined in the tables in sections *C.13.2 Firewall Controller register summary* on page Appx-C-399 and *C.13.3 Firewall Components register summary* on page Appx-C-401 for the Firewall Controller and Firewall Component respectively.
- Locations which are Reserved due to the level of extension implemented in the Firewall Component. For example, the RWE registers when PE.0 is implemented.
- Locations which are Reserved due to the configuration of the Firewall Component. For example, accessing a register of the RWE when the RWE_CTRL.RGN_INDEX refers to a region which is not implemented.

The above does not include the case where a field within a register is Reserved, due to the configuration of a level of extensions implemented in the Firewall Component. In this case the Reserved field returns its Reserved value.

The table below shows the memory layout of a Firewall.

Table C-89 Summary of Firewall memory map

Offset	Component
`0x00_0000`	Firewall Controller
`0x01_0000`	Firewall Component 1/Reserved
`0x02_0000`	Firewall Component 2/Reserved
…	Firewall Component N/Reserved
`0x1F_0000`	Firewall Component 31/Reserved

This section contains the following subsections:

- *C.13.1 Configuration access* on page Appx-C-398.
- *C.13.2 Firewall Controller register summary* on page Appx-C-399.
- *C.13.3 Firewall Components register summary* on page Appx-C-401.

C.13.1 Configuration access

This section describes supported configuration access and configuration access responses.

Supported configuration access

The Firewall must support aligned 32-bit word accesses to the registers of the Firewall.

Any other accesses generate a Configuration Access Error. Accesses to the Firewall registers should be treated as Device-nRnGnE or Device-nRnGE, as defined by *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*, to guarantee that accesses occur in program order and do not cause any unexpected side effects.

Configuration access responses

The following table shows the response generated by the Firewall for configuration access which generate a Configuration Access Error.

Table C-90 Firewall Controller behavior for configuration accesses which generate a Configuration Access Error

Transaction type	FW_ST.ERR	FW_ST.RAZ	Response
Read	0b0	0b0	A non-error read response is generated with the read data set to a value specific to the StreamID
	0b0	0b1	A non-error read response is generated with the read data set to all 0s
	0b1	0b0	An error read response is generated with the read data set to a value specific to the StreamID
	0b1	0b1	An error read response is generated with the read data set to all 0s
Write	0b0	X	A non-error write response is generated
	0b1	X	An error write response is generated

The StreamID specific read data is set by an IMPLEMENTATION DEFINED method.

C.13.2 Firewall Controller register summary

The table below lists the Firewall Controller registers.

Table C-91 Firewall Controller registers

Offset	Short name	Lockable	Save and restore
0x000	*FW_CTRL* on page Appx-C-384	Y	Y
0x004	*FW_ST* on page Appx-C-385	N	N
0x008	Reserved	-	-
0x00C	*FW_SR_CTRL* on page Appx-C-385	Y	Y
0x010	*LD_CTRL* on page Appx-C-368	a	Y
0x014 – 0x0FC	Reserved	-	-
0x100	*PE_CTRL* on page Appx-C-321	Y	Y
0x104	*PE_ST* on page Appx-C-322	N	N
0x108	*PE_BPS* on page Appx-C-323	N	N
0x10C	*RWE_CTRL* on page Appx-C-324	N	Y
0x110	*RGN_CTRL0* on page Appx-C-324	b	c
0x114	*RGN_CTRL1* on page Appx-C-324	b	c
0x118	*RGN_LCTRL* on page Appx-C-325	b	c
0x11C	*RGN_ST* on page Appx-C-326	N	N
0x120	*RGN_CFG0* on page Appx-C-328	b	c
0x124	*RGN_CFG1* on page Appx-C-328	b	c
0x128	*RGN_SIZE* on page Appx-C-329	b	c
0x12C	Reserved	-	-

Table C-91 Firewall Controller registers (continued)

Offset	Short name	Lockable	Save and restore
0x130	*RGN_SIZE* on page Appx-C-329	b	c
0x134	*RGN_TCFG1* on page Appx-C-330	b	c
0x138	*RGN_TCFG2* on page Appx-C-331	b	c
0x13C	Reserved	-	-
0x140	*RGN_MID0* on page Appx-C-332	b	c
0x144	*RGN_MPL0* on page Appx-C-333	b	c
0x148	*RGN_MID1* on page Appx-C-332	b	c
0x14C	*RGN_MPL1* on page Appx-C-333	b	c
0x150	*RGN_MID2* on page Appx-C-332	b	c
0x154	*RGN_MPL2* on page Appx-C-333	b	c
0x158	*RGN_MID3* on page Appx-C-332	b	c
0x15C	*RGN_MPL3* on page Appx-C-333	b	c
0x160-0x17C	Reserved	-	-
0x180	*FE_TAL* on page Appx-C-335	N	N
0x184	*FE_TAU* on page Appx-C-335	N	N
0x188	*FE_TP* on page Appx-C-335	N	N
0x18C	*FE_MID* on page Appx-C-336	N	N
0x190	*FE_CTRL* on page Appx-C-336	N	N
0x194-0x1FC	Reserved		
0x200	*ME_CTRL* on page Appx-C-350	Y	Y
0x204	*ME_ST* on page Appx-C-350	N	N
0x208 – 0x25C	Reserved	-	-
0x260	*EDR_TAL* on page Appx-C-351	N	N
0x264	*EDR_TAU* on page Appx-C-351	N	N
0x268	*EDR_TP* on page Appx-C-351	N	N
0x26C	*EDR_MID* on page Appx-C-352	N	N
0x270	*EDR_CTRL* on page Appx-C-352	N	N
0x274 – 0xCFC	Reserved	-	-
0xD00 - 0xD7C	*FC{0-31}_INT_ST* on page Appx-C-387	N	N
0xD80 – 0xD8C	Reserved	-	-
0xD90	*FW_INT_ST* on page Appx-C-387	N	N
0xD94 – 0xDFC	Reserved	-	-
0xE00-0xE7C	*FC{0-31}_INT_MSK* on page Appx-C-387	Y	Y
0xE80-0xE8C	Reserved	-	-
0xE90	*FW_TMP_TA* on page Appx-C-379	N	N

Table C-91 Firewall Controller registers (continued)

Offset	Short name	Lockable	Save and restore
0xE94	Reserved	-	-
0xE98	*FW_TMP_TP* on page Appx-C-379	N	N
0xE9C	*FW_TMP_MID* on page Appx-C-380	N	N
0xEA0	*FW_TMP_CTRL* on page Appx-C-380	N	N
0xEA4-0xF9C	Reserved		
0xFA0-0xFAC	*FC_CAP{0-3}* on page Appx-C-297	N	N
0xFB0-0xFBC	*FC_CFG{0-3}* on page Appx-C-300	N	Nd
0xFC0	Reserved		
0xFC4	Reserved		
0xFC8	*IIDR* on page Appx-C-388	N	N
0xFCC	*AIDR* on page Appx-C-389	N	N
0xFD0	*PID4* on page Appx-C-389	N	N
0xFD4	*PID5* on page Appx-C-389	N	N
0xFD8	*PID6* on page Appx-C-389	N	N
0xFDC	*PID7* on page Appx-C-390	N	N
0xFE0	*PID0* on page Appx-C-390	N	N
0xFE4	*PID1* on page Appx-C-390	N	N
0xFE8	*PID2* on page Appx-C-390	N	N
0xFEC	*PID3* on page Appx-C-390	N	N
0xFF0	*CID0* on page Appx-C-391	N	N
0xFF4	*CID1* on page Appx-C-391	N	N
0xFF8	*CID2* on page Appx-C-391	N	N
0xFFC	*CID3* on page Appx-C-391	N	N

a) The LD_CTRL register is only lockable when the Lockdown interface is asserted and the LD_CTRL.LOCK field reads as `0b10` or `0b11`.

b) A region is locked under the following conditions:

- When LDE.1 is implemented and LD_CTRL.LOCK is `0b11`
- When LDE.2 is implemented and LD_CTRL.LOCK is `0b11`
- When LDE.2 is implemented and RGN_ST.LOCK, for the region, is `0b1`

c) Only fields which are programmable are required to be saved and restored.

d) The value of FC_CFG2.PROT_SIZE is not required to be saved and restored, however the sampled values of the associated interface are required to.

C.13.3 Firewall Components register summary

The following table summarizes the registers within a Firewall Component, other than Firewall Component 0.

Table C-92 Summary of registers in a Firewall Component

Offset	Short name	Lockable	Save and restore
0x000 – 0x00C	Reserved	-	-
0x010	*LD_CTRL* on page Appx-C-368	a	Y
0x014 – 0x0FC	Reserved	-	-
0x100	*PE_CTRL* on page Appx-C-321	Y	Y
0x104	*PE_ST* on page Appx-C-322	N	N
0x108	*PE_BPS* on page Appx-C-323	N	N
0x10C	*RWE_CTRL* on page Appx-C-324	N	Y
0x110	*RGN_CTRL0* on page Appx-C-324	b	c
0x114	*RGN_CTRL1* on page Appx-C-324	b	c
0x118	*RGN_LCTRL* on page Appx-C-325	b	c
0x11C	*RGN_ST* on page Appx-C-326	N	N
0x120	*RGN_CFG0* on page Appx-C-328	b	c
0x124	*RGN_CFG1* on page Appx-C-328	b	c
0x128	*RGN_SIZE* on page Appx-C-329	b	c
0x12C	Reserved	-	-
0x130	*RGN_TCFG0* on page Appx-C-330	b	c
0x134	*RGN_TCFG1* on page Appx-C-330	b	c
0x138	*RGN_TCFG2* on page Appx-C-331	b	c
0x13C	Reserved	-	-
0x140	*RGN_MID0* on page Appx-C-332	b	c
0x144	*RGN_MPL0* on page Appx-C-333	b	c
0x148	*RGN_MID1* on page Appx-C-332	b	c
0x14C	*RGN_MPL1* on page Appx-C-333	b	c
0x150	*RGN_MID2* on page Appx-C-332	b	c
0x154	*RGN_MPL2* on page Appx-C-333	b	c
0x158	*RGN_MID3* on page Appx-C-332	b	c
0x15C	*RGN_MPL3* on page Appx-C-333	b	c
0x160 –0x17C	Reserved	-	-
0x180	*FE_TAL* on page Appx-C-335	N	N
0x184	*FE_TAU* on page Appx-C-335	N	N
0x188	*FE_TP* on page Appx-C-335	N	N
0x18C	*FE_MID* on page Appx-C-336	N	N
0x190	*FE_CTRL* on page Appx-C-336	N	N
0x194 – 0x1FC	Reserved		
0x200	*ME_CTRL* on page Appx-C-350	Y	Y

Table C-92 Summary of registers in a Firewall Component (continued)

Offset	Short name	Lockable	Save and restore
0x204	*ME_ST* on page Appx-C-350	N	N
0x208– 0x25C	Reserved	-	-
0x260	*EDR_TAL* on page Appx-C-351	N	N
0x264	*EDR_TAU* on page Appx-C-351	N	N
0x268	*EDR_TP* on page Appx-C-351	N	N
0x26C	*EDR_MID* on page Appx-C-352	N	N
0x270	*EDR_CTRL* on page Appx-C-352	N	N
0x274 – 0xF9C	Reserved	-	-
0xFA0 – 0xFAC	*FC_CAP{0-3}* on page Appx-C-297	N	N
0xFB0 – 0xFBC	*FC_CFG{0-3}* on page Appx-C-300	N	Nd
0xFC0 – 0xFFC	Reserved	-	-

a

The LD_CTRL register is only lockable when the Lockdown interface is asserted and the LD_CTRL.LOCK field reads as 0b10 or 0b11.

b

A region is locked under the following conditions:

- When LDE.1 is implemented and LD_CTRL.LOCK is 0b11
- When LDE.2 is implemented and LD_CTRL.LOCK is 0b11
- When LDE.2 is implemented and RGN_ST.LOCK, for the region, is 0b1

c

Only fields which are programmable are required to be saved and restored.

d

The value of FC_CFG2.PROT_SIZE is not required to be saved and restored, however the sampled values of the associated interface are required to.

Appendix D
Revisions

This appendix describes changes between released issues of this book.

It contains the following section:

D.1 Revisions

This appendix describes changes between released issues of this book.

Table D-1 Differences between issue 0000-00 and issue 0000-04

Change	Location	Affects
First release for Beta. ——— **Note** ——— Changes before this release are not recorded as the documentation was under development. ———	-	First release for Beta

Table D-2 Differences between issue 0000-04 and issue 0000-05

Change	Location	Affects
Document restructured: Parts A, B, and C removed; generic contents of Part B "Components" moved to Appendix; implementation-specific content moved to main body.	• *Appendix A Message Handling Unit* on page Appx-A-252 • *Appendix B Interrupt Router* on page Appx-B-274 • *Appendix C Firewall* on page Appx-C-281	r0p0
Updated styling	Throughout	
Updated Additional reading	*Additional reading* on page 11	
Editorial updates to section	*Chapter 1 Overview* on page 1-14	
Description of Host System rationalized and updated.	*1.3.1 Host System* on page 1-18	
Description of External System updated.	*1.3.3 External Systems* on page 1-18	
Updated section	*1.4 Compliance* on page 1-20	
Configuration values and descriptions updated.	• *2.2 Host System configuration* on page 2-24 • *2.3 External System configuration* on page 2-25	
Updated section	*3.2.2 Reset interface* on page 3-32	
Updated section	*3.2 Host CPU-GIC socket interfaces* on page 3-32	
Clock domain figure updated.	*4.2 Internal clocks* on page 4-62	
Updated section	*Chapter 5 Power* on page 5-68	
New section added	*5.3 Voltage domains* on page 5-72	
Power domain figure updated	*5.4 Power domains* on page 5-73	
Reset tree figure added	*6.1 Reset overview* on page 6-92	
Reset request table updated.	*6.1 Reset overview* on page 6-92	
Updated section	*7.2.2 Debug Authorization Access Control Gate (DAACG)* on page 7-103	
Updated section	*7.4.1 Cross Trigger Interface (CTI)* on page 7-111	
Table updated	*7.8 Granular Power Requestor (GPR)* on page 7-119	
Firewall implementation section added	*8.3 Firewall* on page 8-129	
Updates and editorial applied to Firewall appendix.	*Appendix C Firewall* on page Appx-C-281	

Table D-3 Differences between issue 0000-05 and issue 0000-06

Change	Location	Affects
Change confidential to non-confidential	Throughout the manual	r0p0

Table D-4 Differences between issue 0000-06 and issue 0000-07

Change	Location	Affects
Removed extraneous section descriptions, made editorial changes and corrections, added references	Throughout the manual	r1p0 EAC
Updated publications list	*Additional reading* on page 11	
Updated configuration information	*2.4 Host System Firewall configuration* on page 2-26, *2.6 IP configuration* on page 2-29	
Updated clock information	*4.2 Internal clocks* on page 4-62, *4.3 Output clocks* on page 4-67	
Updated power information	*5.2 PPU* on page 5-70, *5.5 Power domain relationship* on page 5-82, *5.6 Power states* on page 5-83	
Updated debug information	*7.6 System Trace Macrocell* on page 7-116, *7.8 Granular Power Requestor (GPR)* on page 7-119	
Updated interconnect information	*8.3.4 Firewall read response value* on page 8-134, *8.4 StreamID and CPUID* on page 8-144	
Updated Host System peripherals information	*9.4.1 IMPLEMENTATION DEFINED behavior* on page 9-151	
Updated Programmers model information	*11.2.1 Host CPU interrupt map* on page 11-172	
Updated Software sequences information	*12.2 Time domains* on page 12-242	
Updated MHU information	*A.2.3 Ready to Send* on page Appx-A-255, *A.2.4 Interrupts* on page Appx-A-256, *Receiver Frame registers* on page Appx-A-258, *Response Configuration (RESP_CFG)* on page Appx-A-263	
Updated Firewall information	*Region enable* on page Appx-C-310, *Fault entry acknowledgement* on page Appx-C-315, *Fault entry and power* on page Appx-C-315, *Fault Entry Control (FE_CTRL)* on page Appx-C-336, *Output transaction security property* on page Appx-C-359, *Output transaction instruction property* on page Appx-C-359, *C.11.1 Protection Extension* on page Appx-C-373, *Tamper report* on page Appx-C-378, *Access Error* on page Appx-C-382, *Programming Error* on page Appx-C-382, *Implementation Identification (IIDR) register* on page Appx-C-388, *Peripheral ID 3 (PID3) register* on page Appx-C-390	