Arm® Corstone™ SSE-123 Example Subsystem

Revision: r0p0

Technical Reference Manual

Non-Confidential

Issue 02
101370_0000_02_en

Arm® Corstone™ SSE-123 Example Subsystem
Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-00	22 March 2019	Non-Confidential	First release for r0p0
0000-01	10 April 2020	Non-Confidential	Second release for r0p0
0000-02	30 June 2022	Non-Confidential	Third release for r0p0

Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Additional Notices

Some material in this document is based on IEEE 754-2008 IEEE Standard for Binary Floating-Point Arithmetic. The IEEE disclaims any responsibility or liability resulting from the placement and use in the described manner.

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.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com.

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

This document includes language that can be offensive. We will replace this language in a future issue of this document.

To report offensive language in this document, email terms@arm.com.

Contents

1. Introduction

1.1 Product revision status

The r*x*p*y* identifier indicates the revision status of the product described in this manual, for example, r*1*p*2*, where:

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

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

1.2 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 Arm® Corstone™ SSE-123 Example Subsystem.

1.3 Conventions

The following subsections describe conventions used in Arm documents.

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: developer.arm.com/glossary.

Typographic conventions

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. Signal names. Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace bold`	Language keywords when used outside example code.

Convention	Use
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<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	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.

Caution

Recommendations. Not following these recommendations might lead to system failure or damage.

Warning

Requirements for the system. Not following these requirements might result in system failure or damage.

Danger

Requirements for the system. Not following these requirements will result in system failure or damage.

Note

An important piece of information that needs your attention.

Tip

A useful tip that might make it easier, better or faster to perform a task.

Remember

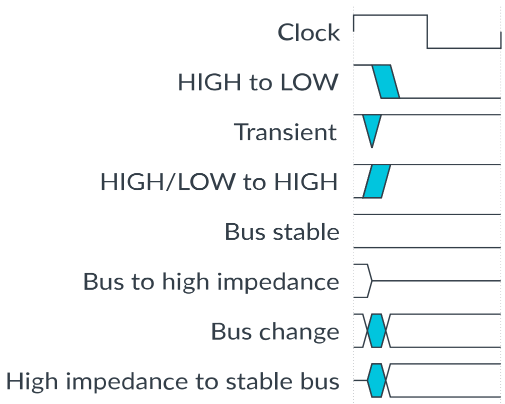
A reminder of something important that relates to the information you are reading.

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 1-1: Key to timing diagram conventions



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.

1.4 Additional reading

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

Table 1-2: Arm publications

Document name	Document ID	Licensee only
Arm® SSE-123 Example Subsystem Technical Overview	101371	No
Arm®v8-M Architecture Reference Manual	DDI 0553	No
Arm® AMBA® 5 AHB Protocol Specification Version: 2.0	IHI 0033	No
AMBA® APB Protocol Specification Version: 2.0	IHI 0024	No

Document name	Document ID	Licensee only
Arm® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2	IHI 0031	No
AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces	IHI 0068	No
Arm® Power Policy Unit Version 1.1 Architecture Specification	DEN 0051E	No
Arm® Cortex®-M23 Processor Technical Reference Manual	DDI 0550	No
Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual	DDI 0571	No
Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual	101150	No
Arm® CoreSight™ Architecture Specification	IHI 0029	No
Arm® CoreSight™ ETM-M23 Technical Reference Manual	DDI 0563	No
Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual	100806	No
Arm® CoreLink™ CG092 AHB Flash Cache Technical Reference Manual	DDI 0569	No
Arm® Clock Controller Architecture Specification Version 1.0	DEN 0052	Yes
Arm® Power Control System Architecture Specification Version 2.0	DEN 0050	Yes
Arm® SSE-123 Example Subsystem Configuration and Integration Manual	101372	Yes
Arm® SSE-123 Example Subsystem Release Note	PJDOC-1779577084-12680	Yes
Arm® SSE-123 Example Subsystem Analysis Report	PJDOC-1779577084-28939	Yes
Arm® SSE-123 Example Subsystem Verification Summary Report	PJDOC-1779577084-28938	Yes
Arm Platform Security Architecture Trusted Base System Architecture for Armv6-M, Armv7-M and Armv8-M	DEN 0083	Yes
Arm® Cortex®-M23 Processor Integration and Implementation Manual	DIT 0062	Yes
Arm® CoreLink™ CG092 AHB Flash Cache Configuration and Integration Manual	DIT 0065	Yes
Arm® CoreLink™ PCK-600 Power Control Kit Configuration and Integration Manual	101151	Yes
Arm® CoreLink™ SIE-200 System IP for Embedded Configuration and Integration Manual	DIT 0067	Yes

Note

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

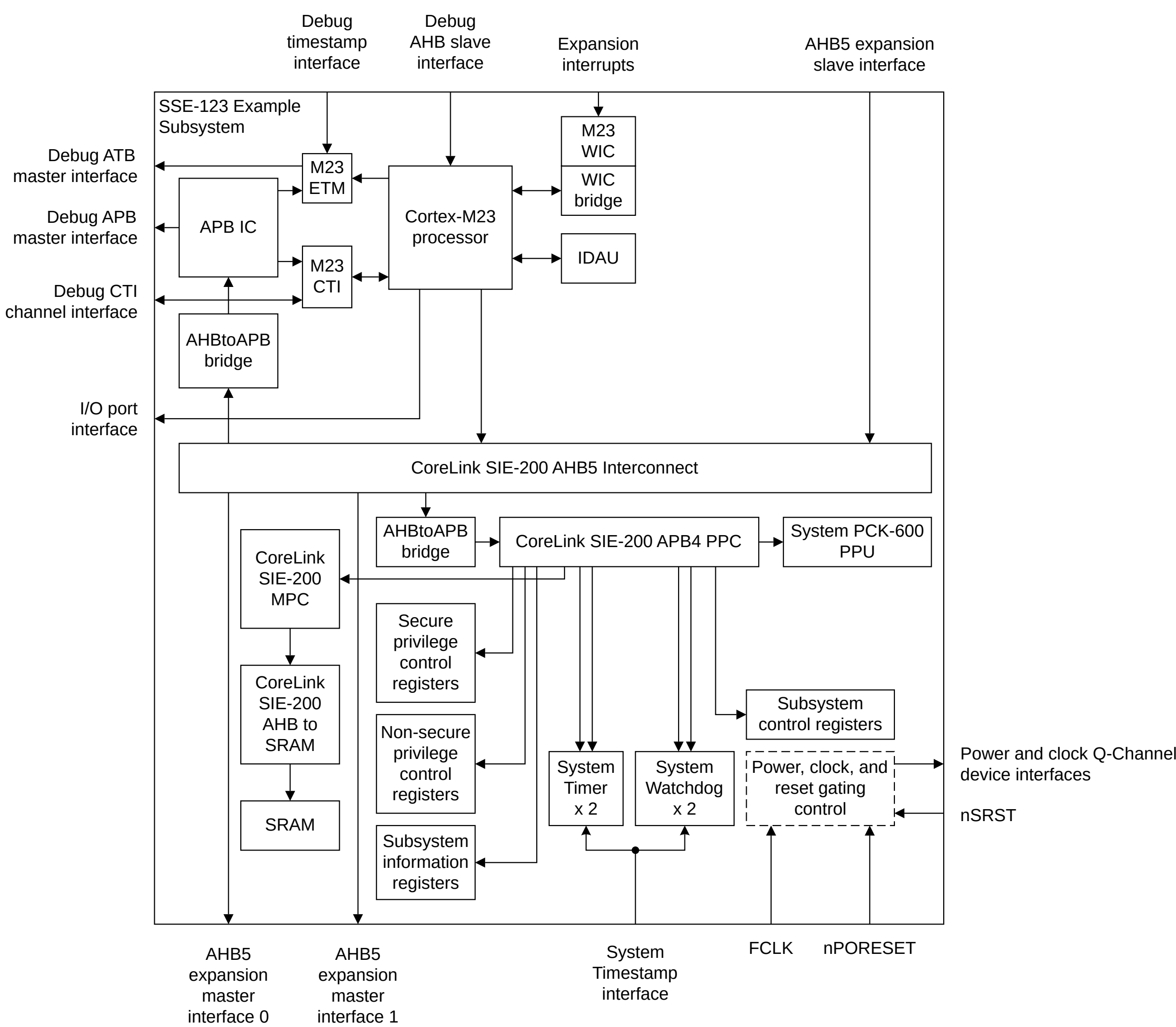
Adobe PDF reader products can be downloaded at http://www.adobe.com

2. SSE-123 Example Subsystem

The SSE-123 Example Subsystem integrates a subsystem of key Arm components that implement core functionality of a system targeting *Internet of Things* (IoT) *System on Chip* (SoC) designs.

The following figure shows a block diagram of the SSE-123 Example Subsystem.

Figure 2-1: SSE-123 Example Subsystem block diagram



The block diagram shows all the key integrated components and interfaces.

2.1 About IoT System on Chip implementations

The SSE-123 Example Subsystem must be extended to create an IoT SoC. A complete system typically contains the following components:

Compute subsystem

The compute subsystem consists of a single Cortex®-M23 processor and associated bus, debug, controller, peripherals, and interface logic supplied by Arm.

Reference system memory and peripherals

SRAM is part of the SSE-123 Example Subsystem, but an SoC requires extra memory, control, and peripheral components beyond the minimum subsystem components. Flash memory, for example, is not provided with the SSE-123 Example Subsystem.

Communication interface

The endpoint must have some way of communicating with other nodes or masters in the system. This interface could be WiFi, Bluetooth, or a wired connection.

Sensor or control component

To be useful as an endpoint, the reference design is typically extended by adding sensors or control logic such as temperature input or motor control output.

Software development environment

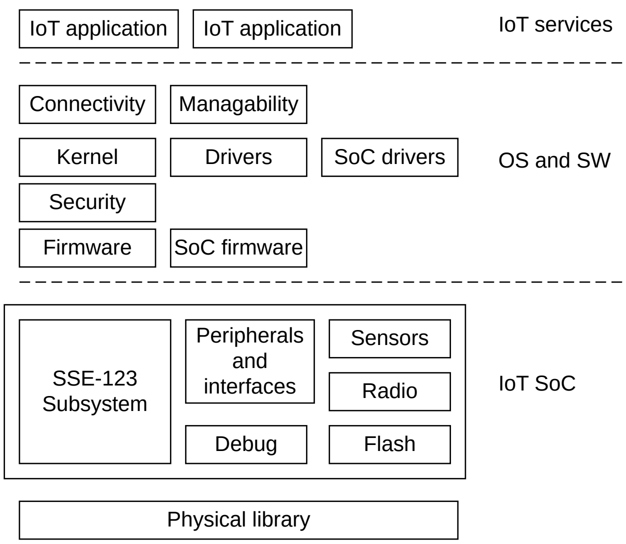
Arm provides a complete software development environment which includes the Arm® Mbed™ operating system, Arm or GNU (GCC) compilers and debuggers, and firmware.

Custom peripherals typically require corresponding third-party firmware that can be integrated into the software stack.

2.1.1 IoT hardware and software

The following figure shows a block diagram of the hardware and software in an IoT system.

Figure 2-2: Hardware and software solution



2.2 Compliance

The SSE-123 Example Subsystem complies with, or implements, the specifications that this section describes. This document complements architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources.

2.2.1 Arm architecture

The Cortex®-M23 processor in the subsystem implements the Armv8-M Baseline Architecture with Security Extension.

See the *Arm®v8-M Architecture Reference Manual*.

2.2.2 Security architecture

The SSE-123 is designed to facilitate implementation of a TBSA-M compliant system.

See the *Arm® Platform Security Architecture - Trusted Base System Architecture for Armv8-M*.

2.2.3 Interrupt controller architecture

The SSE-123 implements Arm® *Nested Vector Interrupt Controller* (NVIC) and Arm® *Wakeup Interrupt Controller* (WIC).

See the *Arm® Cortex®-M23 Processor Technical Reference Manual*.

2.2.4 Advanced Microcontroller Bus Architecture (AMBA®)

The SSE-123 implements the following interface protocol architectures:

- *Advanced High-Performance Bus 5* (AHB5). See the *AMBA® 5 AHB Protocol Specification*.
- *Advanced Peripheral Bus 4* (APB4). See the *AMBA® APB Protocol Specification Version: 2.0*.
- *Low-Power Interface* (LPI), Q-Channel, and- P-Channel. See the *AMBA® Low Power Interface Specification*.

2.2.5 Debug architecture

The SSE-123 implements the Arm® *Debug Interface Architecture 5* (ADIv5)-compliant debug interfaces.

See the *Arm® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*.

2.2.6 Power control architecture

The SSE-123 implements the framework for system power control that the Arm® *Power Control System Architecture* (PCSA) Version 2.0 specification defines.

See the *Arm® Power Control System Architecture Specification Version 2.0*.

2.3 Features of SSE-123

The SSE-123 Example Subsystem provides the following features:

- A Cortex®-M23 processor, including Armv8-M Security Extensions
- A single bank of system SRAM
- CoreLink™ SIE-200 System IP for Embedded:
  - AHB5 bus matrix
  - *Memory Protection Controller* (MPC)
  - *Peripheral Protection Controller* (PPC)
  - AHB5 to APB4 bridge

    - AHB5 to SRAM controller
- CoreLink™ PCK-600 Power Control Kit:
    - *Power Policy Unit* (PPU)
    - Clock controller
    - Low-Power Distributor Q-Channel (LPD-Q)
- *Implementation Defined Attribution Unit* (IDAU)
- Cortex®-M23 processor *Wakeup Interrupt Controller* (WIC)
- System Timer and Watchdog
- System Control and Security Control Registers
- Optional Cortex®-M23 processor Debug components:
    - *Embedded Trace Macrocell* (ETM)
    - *Cross Trigger Interface* (CTI)
    - Debug APB interconnect

2.4 Configurable options

The SSE-123 Example Subsystem is highly configurable and provides configuration options for features of the design.

2.4.1 Subsystem configuration options

This section defines the configuration options for the SSE-123 Example Subsystem.

The following table shows the subsystem configuration options.

Table 2-1: Subsystem configuration options

Feature	Configurable options
SRAM size	8KB to 16MB.
SRAM MPC block size	32 bytes to 1MB.
Debug support	Present or absent.
Debug trace	Present or absent. Debug support must also be present for debug trace.
Single-Cycle I/O port	Present or absent.
Memory retention power state	Present or absent.
Interrupts	0-224 expansion interrupt lines are present.
Wakeup interrupt sources	2-242 interrupt lines are used as wakeup sources: **2** Only *Non-Maskable Interrupt* (**NMI**) and **RXEV** are supported. **3-18** **NMI**, **RXEV**, and select internal interrupts are supported. **19-242** **NMI**, **RXEV**, internal interrupts and select expansion interrupts are supported.

Feature	Configurable options
Interrupt disables	Each expansion interrupt line can be independently enabled or disabled.
Interrupt latency	21-255 number of cycles between pending interrupt and vector fetch from the Cortex®-M23 processor.
Vector Table Offset Register (VTOR) initial reset value	Address at reset for the Cortex®-M23 processor VTOR_NS.
Security control expansion	Security control signals and security violation interrupts for the following expansion devices: **0-16** *Memory Protection Controller* (MPC). **0-4** AHB *Peripheral Protection Controller* (PPC) with 0-16 peripherals. **0-4** APB *Peripheral Protection Controller* (PPC) with 0-64 peripherals. **0-16** *Master Security Controller* (MSC). **0-16** Bridge error interrupts.
External bus access control	Blocking or not blocking.

2.4.2 Processor configuration options

The SSE-123 Example Subsystem supports a range of configurable options for the Cortex®-M23 processor.

The following table shows the configurable choices for the Cortex®-M23 processor and includes the options that are constrained for the SSE-123 Example Subsystem.

Table 2-2: Integrated IP configuration options

Feature	Configurable options
Non-secure *Memory Protection Unit* (MPU)	8, 12, 16 regions.
Secure MPU	8, 12, 16 regions.
Security Attribution Unit (SAU)	Fixed 8 regions.
SysTick timers	0, 1, or 2 timers can be present. If only one timer is present, it is configurable by software whether it is Secure or Non-secure.
Vector Table Offset Register (VTOR)	Fixed present.
Reset all registers	Present or absent.
Multiplier	Fast, one cycle, or slow, 32 cycles.
Divider	Fast, 17 cycles, or slow, 34 cycles.
Interrupts	Fixed to 16 + subsystem expansion interrupts.
Instruction fetch width	16-bit only or 32-bit.
Single-Cycle I/O port	Present or absent. Must be present for subsystem Single-Cycle I/O port.
Architectural clock gating	Present or absent.
Data endianness	Fixed little-endian.
Halting debug support	Present or absent. Must be present for subsystem debug support.
Wake-up interrupt controller	Fixed present.
Number of breakpoint comparators	0, 1, 2, 3, 4.

Feature	Configurable options
Number of watchpoint comparators	0, 1, 2, 3, 4.
Cross Trigger Interface (CTI)	Present or absent. Must be present for subsystem debug support.
Micro Trace Buffer (MTB)	Fixed absent.
Embedded Trace Macrocell (ETM)	Present or absent. Must be present for subsystem debug trace.
JTAGnSW debug protocol	Selects between JTAG or Serial-Wire interfaces for the DAP.
Multi-drop support for serial wire	Present or absent.
Slave port support for AHB DAP	When set, include slave port support for any AHB DAP implementation. Otherwise, support only the low area DAP.

2.5 Product documentation

This section describes the SSE-123 product documentation in relation to the design flow.

2.5.1 Documentation

The SSE-123 Example Subsystem documentation is as follows:

Technical Overview

The *Technical Overview* (TO) provides a high-level overview of the SSE-123 Example Subsystem:

- Hardware.
- Software.

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the SSE-123 Example Subsystem. It is required at all stages of the design flow. The choices that are made in the design flow can mean that some behaviors that are described in the TRM are not relevant. If you are programming the SSE-123, then contact:

- The implementer to determine:
    - The build configuration of the implementation.
    - The integration, if any, that was performed before implementing the SSE-123.
- The integrator to determine the pin configuration of the device that you are using.

Configuration and Integration Manual

The *Configuration and Integration Manual* (CIM) describes:

- The available build configuration options and related issues in selecting them.
- Guidelines on how to integrate the SSE-123 Example Subsystem into an SoC.
- The SSE-123 Integration component, providing examples of integration with Arm® eFlash and debug products.

- The processes to sign off the configuration, integration, and physical implementation of the design.

The CIM is a confidential book that is only available to licensees.

Verification Summary Report

The *Verification Summary Report* (VSR) describes:

- An overview of verification performed on the SSE-123 Example Subsystem.
- The verification quality definition for the subsystem.
- Configurations of the subsystem verified.
- A summary of verification results.

The VSR is a confidential book that is only available to licensees.

Subsystem Analysis Report

The *Subsystem Analysis Report* (SAR) describes:

- Performance characteristics of the SSE-123 Example Subsystem.
- Processor performance analysis and benchmark results.
- Performance analysis of memory system bandwidth and latency.

The SAR is a confidential book that is only available to licensees.

2.6 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 First release.

3. Functional description

The following sections describe the functionality of the SSE-123 Example Subsystem.

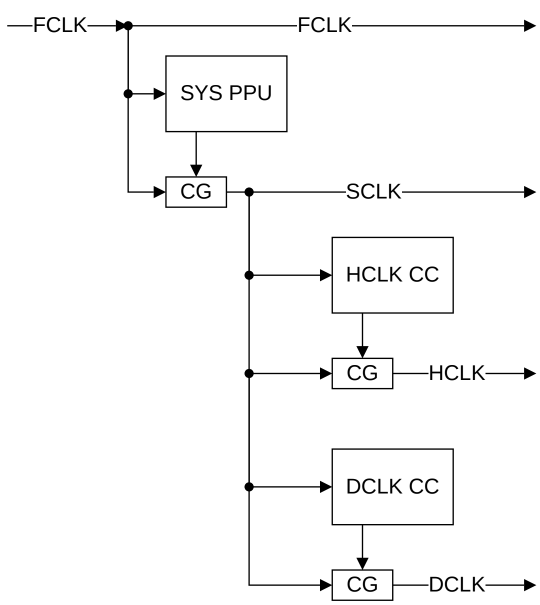
3.1 Clocks

This section describes the clocks in the SSE-123 Example Subsystem.

Top-level overview

The following figure shows the SSE-123 Example Subsystem clock structure, including *Clock Controllers* (CCs) and *Power Policy Units* (PPUs) that control the *Clock Gates* (CGs) in the subsystem.

Figure 3-1: Top-level overview of SSE-123 Example Subsystem clocks



All clocks in the subsystem are synchronous to a single input clock, **FCLK**.

The **FCLK** signal clocks the following:

- Subsystem control logic.
- Cortex®-M23 *Wakeup Interrupt Controller* (WIC).
- Subsystem timers.
- Subsystem watchdogs.

The subsystem considers **FCLK** to be free running, but can be gated outside the subsystem if it is in the CSS.OFF power state. See 3.3.3 Subsystem power states on page 31.

The following clock is derived from **FCLK**:

SCLK Clocks the Cortex®-M23 *Nested Vector Interrupt Controller* (NVIC) and system control logic in PD_SYS. **SCLK** is gated when the subsystem is in CSS.SLEEP power state. See 3.3.3 Subsystem power states on page 31.

The following clocks are derived from **SCLK** and support dynamic clock gating:

HCLK Clocks the Cortex®-M23 and subsystem interconnect.
DCLK Clocks the subsystem debug components.

3.1.1 Clock control

This section describes how clocks are controlled in the SSE-123 Example Subsystem.

FCLK

Control of **FCLK** is expected to be integrated outside the subsystem. When the system is in operation, it is expected that **FCLK** is free-running. The frequency of **FCLK** can be switched during operation. See 3.1.4 FCLK dynamic clock switching on page 24.

SCLK

The power state determines the clock gating of **SCLK**. The **SCLK** clock is enabled during the ON power mode. Otherwise, the **SCLK** clock is gated. **SCLK** is provided as an output from the subsystem to enable customers to include devices in the clock domain.

HCLK

A Q-Channel clock controller controls the dynamic clock gating of **HCLK**. The **HCLK** clock is gated only when all devices within the clock domain indicate no activity and accept a request for quiescence. **HCLK** is provided as an output from the subsystem to enable customers to include devices in the clock domain. To enable safe dynamic gating of **HCLK**, connect the **HCLK** Q-Channel interface to all devices outside the subsystem that use **HCLK**.

DCLK

A Q-Channel clock controller controls the dynamic clock gating of **DCLK**. The **DCLK** clock is gated only when all devices within the clock domain indicate no activity and accept a request for quiescence. **DCLK** is provided as an output from the subsystem to enable customers to include devices in the clock domain. To enable safe dynamic gating of **DCLK**, connect the **DCLK** Q-Channel interface to all devices outside the subsystem that use **DCLK**.

3.1.2 External wakeup

This section describes the methods that you can use to wake gated clocks.

You can wake gated clocks in the SSE-123 Example Subsystem using the following mechanisms:

Clock QACTIVE

Raising a clock **QACTIVE** signal on the relevant clock domain expansion Q-Channel interface causes the clock controller to ungate the respective clock.

Power QACTIVE

Raising a power **QACTIVE** signal on the relevant power domain expansion Q-Channel interface causes the *Power Policy Unit* (PPU) to transition to an ON state. This transition ungates the required clocks.

Wakeup Interrupt Controller

The *Wakeup Interrupt Controller* (WIC) operates on **FCLK** to enable it to detect an interrupt and wake the processor from deep sleep. This operation ungates the **SCLK** and **HCLK** domains.

3.1.3 Clock domain expansion

All generated clocks are exported from the subsystem to enable you to add logic in any clock domain.

See A. Signal descriptions on page 88 for information about the clock that is associated with each external interface.

Expansion Q-Channel interfaces are provided for each of the following dynamic clock gated domains:

- **HCLK**.
- **DCLK**.

You must integrate these interfaces with components that are external to the subsystem to ensure that the devices are idle before the clocks are gated. See A.3 Clock control interface signals on page 88.

3.1.4 FCLK dynamic clock switching

The SSE-123 Example Subsystem supports dynamic switching of the frequency of the **FCLK** signal.

Dynamic frequency switching enables the **FCLK** that is supplied to the subsystem to be switched from a high frequency clock to a low frequency clock. This can enable dynamic power saving by switching off high frequency clock generation logic, such as a *Phase Locked Loop* (PLL). The **FCLKSTATEREQn** interface indicates when the system is requesting a high frequency clock.

The FCLK_DCS_ENABLE bits in the 4.2.2.5 CLOCK_CTRL, Clock Control register on page 52 control dynamic clock switching.

The following table shows the **FCLKSTATEREQn** signal in various system power states. See 3.3.3 Subsystem power states on page 31.

Table 3-1: FCLK dynamic clock switching states

Subsystem power state	Cortex®-M23 processor core sleep state	FCLK_DCS_ENABLE	FCLKSTATEREQn
CSS.ON	M23.RUNNING	`2'b00`	LOW
CSS.ON	M23.SLEEPING	`2'b00`	LOW
CSS.ON	M23.DEEPSLEEP	`2'b00`	LOW
CSS.SLEEP	M23.DEEPSLEEP	`2'b00`	LOW
CSS.ON	M23.RUNNING	`2'b01`	LOW
CSS.ON	M23.SLEEPING	`2'b01`	LOW
CSS.ON	M23.DEEPSLEEP	`2'b01`	HIGH
CSS.SLEEP	M23.DEEPSLEEP	`2'b01`	HIGH
CSS.ON	M23.RUNNING	`2'b10`	LOW
CSS.ON	M23.SLEEPING	`2'b10`	LOW
CSS.ON	M23.DEEPSLEEP	`2'b10`	LOW
CSS.SLEEP	M23.DEEPSLEEP	`2'b10`	HIGH

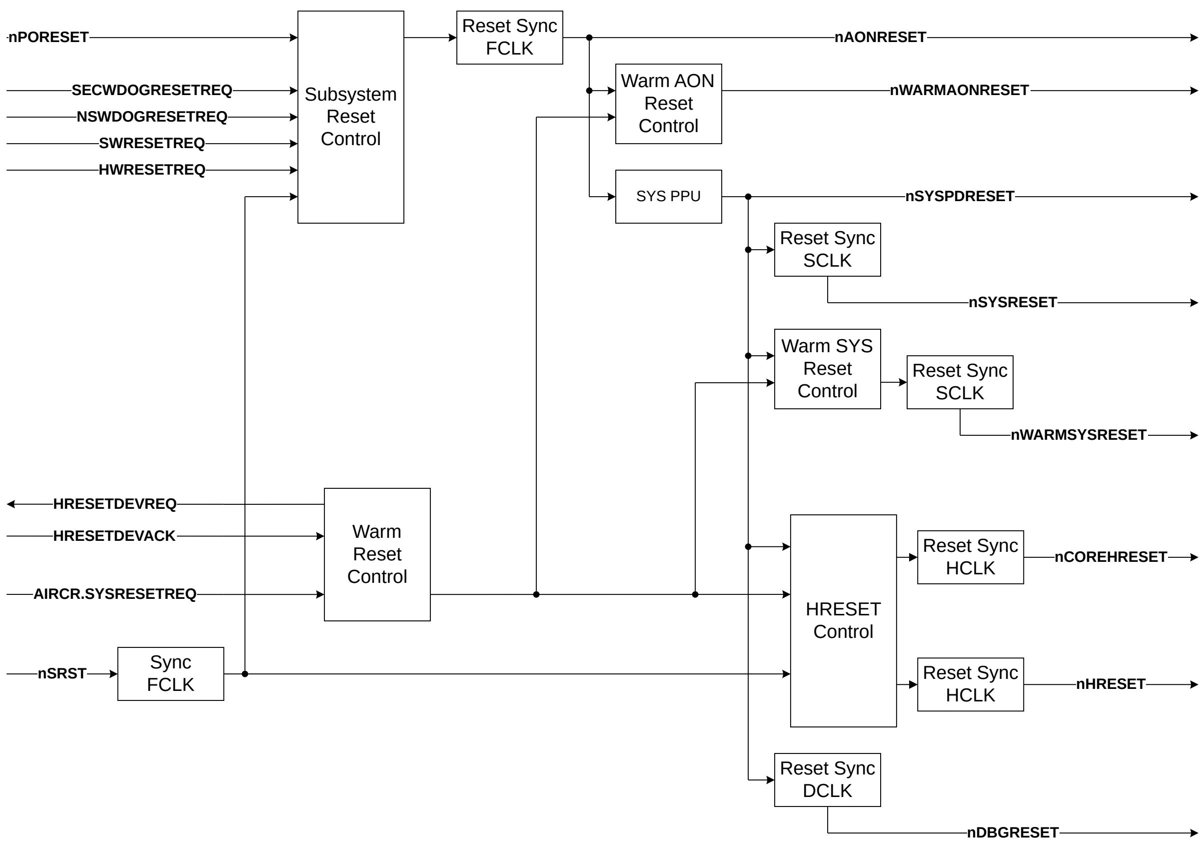
3.2 Resets

This section describes the reset signals in the SSE-123 Example Subsystem.

Top-level overview

The following figure shows the SSE-123 Example Subsystem generated resets, and associated reset control and request signals.

Figure 3-2: SSE-123 Example Subsystem reset structure



The SSE-123 Example Subsystem contains a single reset input, **nPORESET**, that generates all internal resets.

The **nPORESET** reset signal is used for logic in the **PD_AON** power domain of the subsystem. The **nPORESET** reset signal causes all devices in the subsystem to be reset. The SSE-123 Example Subsystem expects that **nPORESET** release is synchronous to **FCLK**.

3.2.1 Internally generated resets

This section describes the internally generated resets in the SSE-123 Example Subsystem.

The SSE-123 Example Subsystem contains the following internally generated resets:

nAONRESET

The **nAONRESET** reset signal is a reset that the SYS reset control generates for all logic in the **PD_AON** power domain, except for the 4.2.2.6 RESET_SYNDROME, Reset Syndrome register on page 53 and 4.2.2.7 RESET_MASK, Reset Mask register on page 54.

This domain resets the following:

- *Power Policy Unit* (PPU).
- System timers.
- System watchdogs.

nWARMAONRESET

The **nWARMAONRESET** reset signal is the reset for subsystem control registers in the **PD_AON** that must be reset when an AIRCR.SYSRESETREQ request is raised. See 4.2.1 Subsystem information registers block on page 45 for information about the registers that this reset affects.

nSYSPDRESET

The **nSYSPDRESET** reset signal is a reset that SYS PPU generates for all logic in **PD_SYS** power domain.

nSYSRESET

The **nSYSRESET** reset signal is a system domain reset for all non-debug logic in the **PD_SYS** power domain. **nSYSRESET** is a reset for the Cortex®-M23 processor system domain, including Cortex®-M23 processor *Nested Vector Interrupt Controller* (NVIC).

nWARMSYSRESET

Reset for subsystem clock control logic in **PD_SYS** that requires reset when a AIRCR.SYSRESETREQ is raised.

nHRESET

The **nHRESET** reset signal is a reset for the following:

- Interconnect.
- *Peripheral Protection Controller* (PPC).
- *Memory Protection Controller* (MPC).
- *Static Random Access Memory* (SRAM).

nCOREHRESET

The **nCOREHRESET** reset signal is a reset for the Cortex®-M23 core.

nDBGRESET

The **nDBGRESET** reset signal is a reset for the Cortex®-M23 debug domain.

3.2.2 Reset requests

The SSE-123 Example Subsystem responds to a range of reset requests that this section describes.

The following table shows the reset request sources for the SSE-123 Example Subsystem.

Table 3-2: SSE-123 Example Subsystem subsystem requests

Name	Source	Description
nSRST	Input signal	You can connect this reset request to the debugger physical interface. It causes the logic in the subsystem to be reset and holds the processor core in reset while **nSRST** is LOW. **Note:** **nSRST** causes reset of the debug logic within the subsystem. For more information about **nSRST**, see the *Arm® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*.
SWRESETREQ	Register	Software reset request from the subsystem control registers block. See 4.2.2.8 SWRESET, Software Reset register on page 55.
HWRESETREQ	Input signal	Hardware reset request enables components that are external to the subsystem to request reset.
SECWDOGRESETREQ	Secure watchdog	The Secure watchdog can request a reset, after a pre-programmed period has elapsed. See 3.8.2 Subsystem watchdogs on page 40.
NSWDOGRESETREQ	Non-secure watchdog	The Non-secure watchdog can request a reset, after a pre-programmed period has elapsed. See 3.8.2 Subsystem watchdogs on page 40. Software can choose to unmask this reset request by writing to RESET_MASK.NSWD_EN. By default, this reset request is masked. See 4.2.2.7 RESET_MASK, Reset Mask register on page 54.
AIRCR.SYSRESETREQ	Cortex®-M23	Cortex®-M23 processor system reset request. When HIGH, indicates that program code or a debugger writing to the AIRCR.SYSRESETREQ bit has requested a reset. This bit enables software or a debugger to request reset of the core. Software can choose to unmask this reset request by writing to the RESET_MASK.SYSRESETREQ_EN bit. By default, this reset request is masked. See 4.2.2.7 RESET_MASK, Reset Mask register on page 54.

You can mask some of the requests by programming software registers. See 4.2.2.7 RESET_MASK, Reset Mask register on page 54 for more information.

3.2.3 Reset control

This section describes how to control resets in the SSE-123 Example Subsystem.

The figure in 3.2 Resets on page 25 shows the following reset control blocks that control the resets resulting from the requests that 3.2.2 Reset requests on page 27 describes.

- Subsystem reset control:
    - Controls the reset of **nAONRESET**.
    - Any of the following reset requests causes the resets to be asserted until the request is cleared:
        - **nSRST** falling edge.
        - **SWRESETREQ**.

        - **HWRESETREQ**.
        - **SECWDOGRESETREQ**.
        - **NSWDOGRESETREQ**.
- Warm reset control:
    - Manages generation of reset requests for:
        - Warm AON reset control.
        - Warm SYS reset control.
        - **HRESET** control.
    - A reset request from AIRCR.SYSRESETREQ does not cause immediate reset of **nWARMAONRESET**, **nWARMSYSRESET**, **nHRESET**, and **nCOREHRESET**. Instead, a request is raised to an external interface of the subsystem.
        - **HRESETDEVREQ** is raised to indicate that a reset is to occur for components that are integrated in the **nHRESET** domain.
        - The external system drives **HRESETDEVACK** to indicate that all expansion devices in the **nHRESET** domain can be safely reset.
        - **HRESETDEVREQ** deasserts only after the **nHRESET** domain goes through a reset sequence and then the domain is released from reset.
        - The **HRESETDEVREQ**/**HRESETDEVACK** interface uses a four-phase handshake protocol to enable connection asynchronously to an external reset controller.
- Warm AON reset control:
    - Controls the reset of **nWARMAONRESET**.
    - Any of the following reset requests causes the reset to be asserted until the request is cleared:
        - All system reset control reset requests.
        - All AON reset control reset requests.
        - AIRCR.SYSRESETREQ.
- Warm SYS reset control:
    - Controls the reset of **nWARMSYSRESET**.
    - Any of the following reset requests causes the reset to be asserted until the request is cleared:
        - All system reset control reset requests.
        - All AON reset control reset requests.
        - AIRCR.SYSRESETREQ.
- **HRESET** control:
    - Controls the reset of **nHRESET** and **nCOREHRESET**.
    - Any of the following reset requests causes the resets to be asserted until the request is cleared:
        - All system reset control reset requests.

        - All AON reset control reset requests.
        - AIRCR.SYSRESETREQ.
    - The state of **nSRST** also controls **nHRESET** and **nCOREHRESET**:
        - The falling edge of **nSRST** causes reset of **nHRESET** and **nCOREHRESET**.
        - **nCOREHRESET** is held in reset while **nSRST** is LOW.

3.3 Power management

This section describes the power management in the SSE-123 Example Subsystem.

3.3.1 Voltage domain

The SSE-123 Example Subsystem operates on a single voltage domain, **VSYS**.

3.3.2 Power domains

The SSE-123 Example Subsystem supports power domains to enable subsystem power-saving states.

The SSE-123 Example Subsystem supports the following power domains:

- An always on power domain, PD_AON.
- A switchable power domain, PD_SYS.

The hierarchy of the power domains is as follows:

PD_AON

Contains the following:

- WIC.
- Timers.
- Watchdogs.

PD_AON must be implemented as an always-on power domain.

PD_SYS

Contains the SRAM and most subsystem components, including the following:

- Cortex®-M23 processor.
- Subsystem interconnect.

Controlled by the PCK-600 PPU integrated into the subsystem:

- Supports the following power states:

    - ON.
    - MEM_RET.
    - OFF.
- Supports dynamic power state transitions.

3.3.3 Subsystem power states

This section describes the subsystem power states in the SSE-123 Example Subsystem.

The following table shows the supported power states of the subsystem. The *Power Policy Unit* (PPU) configuration implements the valid power states for the switchable **PD_SYS** domain.

See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual* for information about configuration of the SYS PPU.

Table 3-3: Subsystem power states

Subsystem power state	PD_AON power state	PD_SYS power state
CSS.RUN	ON	ON
CSS.SLEEP	ON	MEM_RET/OFF
CSS.OFF	OFF	OFF

3.3.4 Entering low-power states

This section describes the subsystem power states in the SSE-123 Example Subsystem.

The *Power Policy Unit* (PPU) controls entering low-power states under the following conditions:

Entering CSS.SLEEP

- The processor must be in *deep sleep* to enter low-power mode.
- The WIC bridge must be enabled by setting **WICBRGEN**.
- The system PPU must be programmed to dynamically transition to a low-power state, MEM_RET/OFF.
- Transition only completes if the PPU completes requests to the following devices:
    - WIC bridge is enabled successfully.
    - Processor power down request is accepted.
    - Expansion **PD_SYS** power Q-Channel power down request is accepted.

3.3.5 Wakeup from low-power states

This section describes wakeup from low-power states in the SSE-123 Example Subsystem. The subsystem enters the CSS.OFF state only when **VSYS** is switched OFF. To wake up from the CSS.OFF state, the system must have power that is applied to **PD_AON**, and have **nPORESET** asserted.

Wakeup from the CSS.SLEEP state to the CSS.RUN state can occur from the following reasons:

- The *Wakeup Interrupt Controller* (WIC) detects an interrupt. See 3.3.7 Wakeup Interrupt Controller (WIC) and WIC-bridge on page 32. The following conditions enable an interrupt as a wakeup source:
    - The interrupt must be included as a wakeup source during the configuration of the subsystem.
    - The *Nested Vector Interrupt Controller* (NVIC) must enable the interrupt, and this in turn enables the interrupt in the WIC.

  The **WICSENSE** output shows the interrupt lines enabled as wakeup sources.
- The WIC detects an event. See 3.3.7 Wakeup Interrupt Controller (WIC) and WIC-bridge on page 32.
- Activity that the **PD_SYS** Power Q-Channel interface indicates. See A.9.1 PD_SYS power Q-Channel signals on page 91.
- Debug powerup request by the debug power interface. See A.9.2 Debug power interface signals on page 92.

3.3.6 Power domain expansion

This section describes the power domain expansion in the SSE-123 Example Subsystem. Power control Q-Channel interfaces are provided to enable the integration of devices within the subsystem power domains.

These interfaces must be integrated with expansion devices that are included in the associated power domains to enable safe power state transitions.

The debug power interface enables an external debugger to power up the system when required.

A power status output, **SYSPPUHWSTAT**, is provided from the *Power Policy Unit* (PPU) to indicate the current power state.

3.3.7 Wakeup Interrupt Controller (WIC) and WIC-bridge

This section describes the *Wakeup Interrupt Controller* (WIC) and WIC-bridge in the SSE-123 Example Subsystem.

The Cortex®-M23 processor WIC is integrated in the subsystem to handle interrupts when the processor *Nested Vector Interrupt Controller* (NVIC) is in a low-power sleep mode.

The Cortex®-M23 processor does not natively support a power domain boundary between the WIC and the NVIC. A WIC-bridge is implemented in the subsystem to enable the **PD_SYS** domain to be powered off, and then be woken up by the WIC.

The WIC is enabled when transitioning to the CSS.SLEEP power state. Software controls the WIC-bridge using the following procedures:

Powering down

1. Enable the WIC-bridge using the 4.2.2.13 WICBRGCTRL, WIC Bridge Control register on page 58 to isolate communication between the WIC and the NVIC.
2. Set SCR.SLEEPDEEP to enable the processor to enter the deep sleep state.
3. Configure the SYS PPU to dynamically request the power OFF state when there is no device activity, and then send the processor into the deep sleep mode using one of the following:
    - A WFI instruction.
    - A WFE instruction.
    - A sleep-on-exit function.

Waking up

1. The WIC observes an unmasked interrupt and causes the PD_SYS power domain to wakeup.
2. Disable the WIC-bridge using the 4.2.2.13 WICBRGCTRL, WIC Bridge Control register on page 58 to re-enable the communication between the WIC and the NVIC.
3. Service the interrupt.
4. The pending interrupt is cleared from the WIC.

3.4 Central processor

This section describes the central processor in the SSE-123 Example Subsystem.

The central processor is located in the following domains:

Power domain	**PD_SYS**
Reset domains	**nSYSRESET**, **nCOREHRESET**, and **nDBGRESET**.

The central processor in the subsystem is a single Cortex®-M23 processor. The processor always includes Armv8-M Security Extension to support Arm® TrustZone® for asset protection.

2.4.2 Processor configuration options on page 19 describes the rendering and parameter configuration of the processor.

The following table shows the static configuration signals for the processor.

Table 3-4: Cortex®-M23 processor static configuration signals

Signal name	Tie value	Description
CFGSTCALIB[25:0]	`0x200_0000`	Secure SysTick calibration configuration. No alternative reference clock is provided, and the frequency of the clock arriving at the processor is not computable in hardware: **CFGSSTCALIB[25]** NOREF = HIGH. **CFGSSTCALIB[24]** SKEW = LOW. **CFGSSTCALIB[23:0]** TENMS = `0x00_0000`.
CFGSTCALIBNS[25:0]	`0x200_0000`	Secure SysTick calibration configuration. No alternative reference clock is provided, and the frequency of the clock arriving at the processor is not computable in hardware: **CFGSSTCALIBNS[25]** NOREF = HIGH. **CFGSSTCALIBNS[24]** SKEW = LOW. **CFGSSTCALIBNS[23:0]** TENMS = `0x00_0000`.
CFGSECEXT	1	Armv8-M security support enabled.
INITVTOR[23:0]	Driven by the 4.2.2.10 INITVTOR, Initial Secure Vector Table Offset register on page 56.	Default Secure vector table offset at reset.
INITVTORNS[23:0]	Set by a configuration parameter during integration of the subsystem.	Default Non-secure vector table offset at reset.
IRQLATENCY[7:0]		The Cortex®-M23 processor supports zero jitter interrupt latency for zero wait-state memory. **IRQLATENCY** specifies the minimum number of cycles between an interrupt that becomes pending in the *Nested Vector Interrupt Controller* (NVIC), and the vector fetch for that interrupt being issued on the AHB-Lite interface. For more information, see the *Arm® Cortex®-M23 Processor Technical Reference Manual*.

3.4.1 *Implementation Defined Attribution Unit* (IDAU)

This section describes the *Implementation Defined Attribution Unit* (IDAU) in the central processor of the SSE-123 Example Subsystem.

The subsystem integrates an IDAU that defines the security attributes of the memory map along with the software-programmable *Security Attribution Unit* (SAU).

16 regions are defined from `0x0-0xF` with fixed security level as the table in 4.1 Subsystem memory map on page 43 defines. The final security level of a region is the highest security level of both SAU and IDAU region definitions.

For more information, see the *Arm® Cortex®-M23 Processor Technical Reference Manual*.

3.4.2 Processor interface expansion

This section describes how to expand processor interfaces in the SSE-123 Example Subsystem.

Single cycle I/O interface

The Cortex®-M23 processor implements an optional single-cycle *I/O Port* (IOP). The subsystem can expose this interface to enable the connection of peripherals outside the subsystem.

The IOP has the following features and limitations:

- Low latency access between the processor and tightly coupled peripherals, such as the *General-Purpose I/O* (GPIO).
- Code cannot be executed from the IOP.
- Peripherals that are integrated on the IOP are only accessible from the processor and the debugger.

For more information, see the *Arm® Cortex®-M23 Processor Technical Reference Manual*.

Interrupt expansion interface

The interrupt expansion interface enables external devices to connect interrupts to the subsystem. Expansion interrupts can be configured to wake up the subsystem from low-power states using the *Wakeup Interrupt Controller* (WIC).

3.5 Interconnect

This section describes the interconnect in the SSE-123 Example Subsystem.

The subsystem integrates a single SIE-200 multi-layer AHB5 bus matrix to connect the subsystem components and expansion interfaces. SIE-200 AHB to APB bridges are integrated to cross power and clock domain boundaries.

For more information, see the *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual*.

3.5.1 Interconnect expansion interfaces

This section describes the expansion interfaces in the SSE-123 Example Subsystem that enable you to integrate extra devices.

See 4.1 Subsystem memory map on page 43 for information about the memory regions that these interfaces access.

The subsystem provides the following interfaces:

AHB5 slave expansion interface

Enables extra masters to access the subsystem using the AHB5 interconnect. To ensure the TBSA-M compatibility of extra masters, it might be necessary to add a security wrapper such as the SIE-200 *Master Security Controller* (MSC) to this interface.

See the *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual*.

AHB5 master expansion interface 0

Enables extra slave devices to be added in the code and SRAM memory regions of the address map. To ensure TBSA-M compatibility of extra devices, it might be necessary to add a gating device such as the SIE-200 *Memory Protection Controller* (MPC) to this interface.

See the *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual*.

AHB master expansion interface 1

Enables extra slave devices to be added to the subsystem. To ensure TBSA-M compatibility of extra devices, it might be necessary to add a gating device such as SIE-200 *Peripheral Protection Controller* (PPC) or MPC.

See the *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual*.

3.5.2 Exclusive access support

The SSE-123 Example Subsystem supports the propagation of exclusive access signaling.

There are no slaves within the subsystem that support exclusive access. However, you can add a slave that supports exclusive access outside the subsystem using the AHB expansion interfaces. The processor can make exclusive accesses to the slave that is connected on the master expansion interface.

3.6 Debug

This section describes the debug in the SSE-123 Example Subsystem. The subsystem optionally integrates debug components within the subsystem to provide standard debug interfaces for integration with expansion logic.

There are three levels of debug features that you can select when you configure the subsystem:

Level 0

No debug components, features, or interfaces are available.

Level 1

The Cortex®-M23 processor is configured with a *Debug Access Port* (DAP) that provides access to the core debug system.

The Cortex®-M23 processor *Cross Trigger Interface* (CTI) is included in the subsystem to enable the integration of processor triggers using a CTI.

The debug APB interconnect is included in the subsystem to provide bus access to debug components.

You can extend the debug subsystem by using the top-level debug APB interface.

Level 2

All components, features, and interfaces of level 1 are included.

Also, the Cortex®-M23 processor *Embedded Trace Macrocell* (ETM) is included in the subsystem to enable integration of the processor instruction trace through the ATB Interface.

3.6.1 Debug authentication

This section describes the debug authentication in the SSE-123 Example Subsystem. The subsystem implements the CoreSight™ authentication interface signals.

You can control these CoreSight™ authentication interface signals by using:

- An external interface.
- The following subsystem control registers:
  - 4.2.2.1 SECDBGSTAT, Secure Debug Status register on page 49.
  - 4.2.2.2 SECDBGSET, Secure Debug Set register on page 50.
  - 4.2.2.3 SECDBGCLR, Secure Debug Clear register on page 51.

For definitions of the authentication interface signals, see the *Arm® CoreSight™ Architecture Specification v3.0*.

3.6.2 Debug Access Port (DAP)

This section describes the *Debug Access Port* (DAP) in the SSE-123 Example Subsystem. You can configure the subsystem to export the Cortex®-M23 processor debug slave interface to enable the connection of M23-DAP or CoreSight™ DAP.

For more information, see the *Arm® Cortex®-M23 Processor Technical Reference Manual*.

The Cortex®-M23 processor debug ROM at `0xE00FF000` defines the debug components within the subsystem. For information about the Cortex®-M23 processor ROM table entries, see *Table 7-1* in the *Arm® Cortex®-M23 Processor Technical Reference Manual*.

All the debug components are available in the memory region `0xE000E000-0xE00FFFFF`. For the full debug memory map descriptions, see the table in 4.1 Subsystem memory map on page 43.

3.6.3 Cross triggers

This section describes the cross triggers in the SSE-123 Example Subsystem. You can configure the subsystem to include a Cortex®-M23 processor *Cross Trigger Interface* (CTI). A cross trigger channel interface is provided to enable the integration to a trigger network outside the subsystem.

Examples of integrating trigger networks outside the subsystem include the addition of a CTI or *Cross Trigger Matrix* (CTM).

For more information about the Cortex®-M23 processor CTI, and the mapping of triggers, see *Appendix G* in the *Arm® Cortex®-M23 Processor Integration and Implementation Manual*.

3.6.4 Trace

This section describes the trace in the SSE-123 Example Subsystem. You can configure the subsystem to include the Arm® CoreSight™ ETM-M23. The CoreSight™ ETM-M23 supports instruction trace only. An ATB interface to the *Embedded Trace Macrocell* (ETM) is provided to connect to an external debug system. You can also connect directly to a *Trace Port Interface Unit* (TPIU).

For more information, see the *Arm® CoreSight™ ETM-M23 Technical Reference Manual*.

3.6.5 Debug expansion

This section describes the debug expansion in the SSE-123 Example Subsystem. You can add extra debug components outside the subsystem using the APB *debug expansion interface*.

A debug ROM might be required to discover components that are added in the expansion region, outside the subsystem. Any debug ROM that is added outside the subsystem must direct the debugger to the Cortex®-M23 processor debug ROM within the subsystem.

If no debug components are required outside the subsystem, the debugger can discover the debug components starting at the Cortex®-M23 processor debug ROM.

3.7 Security control

This section describes security control in the SSE-123 Example Subsystem.

The following register blocks manage security control in the subsystem:

- 4.2.3 Secure privilege control registers block on page 59.
- 4.2.4 Non-secure privilege control registers block on page 80.

The subsystem implements the following protection controllers that the Secure and Non-secure privilege control register blocks control:

- *Memory Protection Controller* (MPC) for the subsystem SRAM.
- *Peripheral Protection Controller* (PPC) for the subsystem peripherals.

3.7.1 Peripheral protection controller

This section describes the *Peripheral Protection Controller* (PPC) in the SSE-123 Example Subsystem.

The PPC is located in the following domains:

Power domain **PD_SYS**
Reset domain **nHRESET**.

The SIE-200 APB4 PPC component is integrated to gate access to subsystem APB peripherals. If an access violation occurs when accessing a protected peripheral, an interrupt is raised.

The 4.2.3 Secure privilege control registers block on page 59 and 4.2.4 Non-secure privilege control registers block on page 80 control the settings of the PPC.

3.7.2 Memory protection controller

This section describes the *Memory Protection Controller* (MPC) in the SSE-123 Example Subsystem.

The MPC is located in the following domains:

Power domain **PD_SYS**
Reset domain **nHRESET**.

The SIE-200 AHB5 MPC component is integrated to gate transactions to the SRAM. If the address is protected, a security violation occurs.

The **cfg_init_value** MPC input is tied LOW so that at boot, the SRAM is Secure only access. During operation, software can program the settings of the MPC to enable Non-secure access.

See 4.2.8 SRAM memory protection control on page 85.

The SRAM MPC parameters are set by options that you select during the configuration of the subsystem.

The *Arm® SSE-123 Example Subsystem Configuration and Integration Manual* defines the SRAM MPC parameters.

3.7.3 Security expansion

This section describes the security expansion in the SSE-123 Example Subsystem. The SSE-123 Example Subsystem provides interfaces for the security control of components outside the subsystem.

See A.15 Security control expansion signals on page 102.

These interfaces enable software to manage the security of devices that are added in the expansion region using the 4.2.3 Secure privilege control registers block on page 59 and 4.2.4 Non-secure privilege control registers block on page 80.

The subsystem supports expansion for:

- Interrupt signaling for:
    - 16 *Memory Protection Controller* (MPC) security violation interrupts.
    - Four AHB *Peripheral Protection Controller* (PPC) security violation interrupts.

    - Four APB PPC security violation interrupts.
    - 16 *Master Security Controller* (MSC) security violation interrupts.
    - 16 bridge error interrupts.
- Security gating control for:
    - 64 APB peripheral devices.
    - 16 AHB peripheral devices.
- Privilege gating control for:
    - 64 APB peripheral devices.
    - 16 AHB peripheral devices.

3.8 Peripherals

This section describes the peripherals in the SSE-123 Example Subsystem.

3.8.1 Subsystem Timers

This section describes the System Timers in the SSE-123 Example Subsystem.

The subsystem integrates two System Timers. See B. System time components on page 107.

The timers operate on the **TSVALUEB_SYS** timestamp input. This enables the subsystem to be included in a shared view of time when integrated into a larger system.

Software can configure each timer condition. When a timer condition is met, an interrupt is raised.

The timer interrupts can be used to wake the **PD_SYS** domain but must be configured to do so before powering down **PD_SYS**. Configuration of the timer is only available when the system is in the CSS.RUN state.

3.8.2 Subsystem watchdogs

This section describes the watchdogs in the SSE-123 Example Subsystem.

The subsystem implements a Secure and Non-secure Subsystem Watchdog. See B. System time components on page 107.

The subsystem watchdogs operate on the **TSVALUEB_SYS** timestamp input. This enables the subsystem to be included in a shared view of time when integrated into a larger system.

Each system watchdog has two interrupt levels that are raised when timeout occurs.

The Non-secure system watchdog can raise an initial timeout interrupt to the processor core. A second timeout raises a second, separate interrupt to the processor core. Software can also choose to raise a reset request on the second watchdog timeout that directly resets the system.

The Secure system watchdog can raise an initial timeout *Non-Maskable Interrupt* (NMI) to the processor core. A second timeout raises a reset request that directly resets the system.

3.9 Subsystem SRAM

This section describes the subsystem SRAM in the SSE-123 Example Subsystem.

The subsystem integrates one bank of single port SRAM that supports zero clock cycle latency. An SIE-200 AHB to SRAM component is integrated to connect the SRAM bank in the subsystem.

All accesses to the SRAM are through a SIE-200 AHB5 MPC. The MPC manages security for the SRAM. See 3.7.2 Memory protection controller on page 39.

4. Programmers model

The following sections describe the programmers model for the SSE-123 Example Subsystem.

Access definition

The programmers model follows the system address map rules that the *Arm®v8-M Architecture Reference Manual* defines.

To provide memory blocks and peripherals that can be mapped either as Secure or Non-secure using software, several address regions are aliased as the table in 4.1 Subsystem memory map on page 43 shows. Software can then choose to allocate each memory block, or peripheral, as Secure or Non-secure using protection controllers. The *Implementation Defined Attribution Unit* (IDAU) region column in the table specifies the Security, ID, and *Non-secure Callable* (NSC) settings for each region.

Except when stated, all accesses to unmapped regions of the memory result in bus error responses. An exception to that is when accessing unmapped address space within a region that a peripheral occupies. In this case, the access is *Read-As-Zero and Write-Ignored* (RAZ/WI). Any accesses that result in security violations return RAZ/WI.

The subsystem implements the following features for controlling access to the memory map:

SAU
: Provides software configurable security attribution.

IDAU
: Provides subsystem-defined security attribution.

MPU
: Provides software-configurable memory protection.

MPC
: Controls access to subsystem SRAM.

PPC
: Controls access to subsystem peripherals.

Secure Privilege Control Register Block
: Configuration for IDAU and PPC.

Non-secure Privilege Control Register Block
: Configuration for PPC.

The following definitions are used in this section:

Security

NS	Non-secure access only.
S	Secure access only.
NSC	Non-secure callable access.

NSP	Non-secure privileged access only.
SP	Secure privilege access only.
ALL	All security types have access.

Access

RO	Read-only, and Write-Ignore.
WO	Write-only, and Read-As-Zero.
RW	Read and Write.
RAZ/WI	Read-As-Zero and Write-Ignore.

4.1 Subsystem memory map

This section describes the SSE-123 Example Subsystem memory map.

Access to specific regions is limited to only the processor and debug access interface. Accesses to those regions from any other master or expansion interface return a bus error. This behavior applies to the following regions:

- Processor I/O port.
- All devices in the processor private peripherals regions and debug APB expansion interface, region ID 50-56.

Accesses to reserved regions, that are private to the processor and debug access interface, do not return a bus error, but instead are RAZ/WI.

The following configuration options affect the memory map:

- When the Single-Cycle I/O port feature is not present, 'Processor I/O Port' regions are reserved and access to these regions results in a bus error.
- The NSCCFG configures the NSC values for Secure code regions.
- Programming of the *Memory Protection Controller* (MPC), *Peripheral Protection Controller* (PPC), and/or privilege control register blocks define the privileged and unprivileged accessibility. For information about devices with control of privilege level, see the following:
    - 4.2.3 Secure privilege control registers block on page 59.
    - 4.2.4 Non-secure privilege control registers block on page 80.

The following table shows the assignments of memory regions. The table includes the following information:

- IDAU decoding defining the security attribution of a memory region.
- Device aliasing defines which physical devices are aliased across Secure and Non-secure memory regions.

Note

When the table defines aliasing of expansion interfaces across Secure and Non-secure memory regions, these interfaces are not aliased at the interfaces, but expected to be aliased outside the subsystem. In addition, MPCs and PPCs should be connected for these regions, outside the subsystem, to selectively map memory blocks of peripherals between Secure and Non-secure regions.

Table 4-1: Subsystem memory region descriptions

Region ID	Address		Size	Region description	Alias with region ID	IDAU region values		
	From	To				Security	IDAUID	NSC
1	`0x00000000`	`0x0DFFFFFF`	224MB	AHB5 expansion master interface 0.	4	NS	`0x0`	0
2	`0x0E000000`	`0x0E001FFF`	8KB	Reserved.	-			
3	`0x0E002000`	`0x0FFFFFFF`	32760KB	Reserved.	-			
4	`0x10000000`	`0x1DFFFFFF`	224MB	AHB5 expansion master interface 0.	1	S	`0x1`	CODE NSC
5	`0x1E000000`	`0x1E001FFF`	8KB	Reserved.	-			
6	`0x1E002000`	`0x1FFFFFFF`	32760KB	Reserved.	-			
7	`0x20000000`	`0x20FFFFFF`	16MB	Subsystem SRAM.	10	NS	`0x2`	0
8	`0x21000000`	`0x27FFFFFF`	112MB	Reserved.	-			
9	`0x28000000`	`0x2FFFFFFF`	128MB	AHB5 expansion master interface 0.	-			
10	`0x30000000`	`0x30FFFFFF`	16MB	Subsystem SRAM.	7	S	`0x3`	RAM NSC
11	`0x31000000`	`0x37FFFFFF`	112MB	Reserved.	-			
12	`0x38000000`	`0x3FFFFFFF`	128MB	AHB5 expansion master interface 0.	-			
13	`0x40000000`	`0x40000FFF`	4KB	Subsystem timer 0.	25	NS	`0x4`	0
14	`0x40001000`	`0x40001FFF`	4KB	Subsystem timer 1.	26			
15	`0x40002000`	`0x4001FFFF`	120KB	Reserved.	-			
16	`0x40020000`	`0x40020FFF`	4KB	Subsystem Information Registers.	28			
17	`0x40021000`	`0x4007FFFF`	380KB	Reserved.	-			
18	`0x40080000`	`0x40080FFF`	4KB	Non-secure privilege control registers.	-			
19	`0x40081000`	`0x40082FFF`	8KB	Non-secure subsystem watchdog.	-			
20	`0x40083000`	`0x40087FFF`	20KB	Reserved.	-			
21	`0x40088000`	`0x4008BFFF`	16KB	Reserved.	-			
22	`0x4008C000`	`0x400EFFFF`	400KB	Reserved.	-			
23	`0x400F0000`	`0x400FFFFF`	64KB	Processor I/O port.	40			
24	`0x40100000`	`0x4FFFFFFF`	255MB	AHB5 expansion master interface 1.	-			
25	`0x50000000`	`0x50000FFF`	4KB	Subsystem timer 0.	13	S	`0x5`	0
26	`0x50001000`	`0x50001FFF`	4KB	Subsystem timer 1.	14			
27	`0x50002000`	`0x5000FFFF`	120KB	Reserved.	-			
28	`0x50020000`	`0x50020FFF`	4KB	Subsystem information registers.	17			
29	`0x50021000`	`0x50021FFF`	4KB	Subsystem control registers.	-			
30	`0x50022000`	`0x50022FFF`	4KB	System *Power Policy Unit* (PPU).	-			
31	`0x50023000`	`0x50026FFF`	16KB	Reserved.	-			
32	`0x50027000`	`0x50027FFF`	4KB	Reserved.	-			

Region ID	Address		Size	Region description	Alias with region ID	IDAU region values		
	From	To				Security	IDAUID	NSC
33	0x50028000	0x5007FFFF	352KB	Reserved.	-			
34	0x50080000	0x50080FFF	4KB	Secure privilege control registers.	-			
35	0x50081000	0x50082FFF	8KB	Secure subsystem watchdog.	-			
36	0x50083000	0x50083FFF	4KB	SRAM memory protection control.	-			
37	0x50084000	0x50087FFF	16KB	Reserved.	-			
38	0x50088000	0x5008BFFF	16KB	Reserved.	-			
39	0x5008C000	0x500EFFFF	400KB	Reserved.	-			
40	0x500F0000	0x500FFFFF	64KB	Processor I/O port.	23			
41	0x50100000	0x5FFFFFFF	255MB	AHB5 expansion master interface 1.	-			
42	0x60000000	0x6FFFFFFF	256MB	AHB5 expansion master interface 0.	-	NS	0x6	0
43	0x70000000	0x7FFFFFFF	256MB		-	S	0x7	0
44	0x80000000	0x8FFFFFFF	256MB	AHB5 expansion master interface 1.	-	NS	0x8	0
45	0x90000000	0x9FFFFFFF	256MB		-	S	0x9	0
46	0xA0000000	0xAFFFFFFF	256MB		-	NS	0xA	0
47	0xB0000000	0xBFFFFFFF	256MB		-	S	0xB	0
48	0xC0000000	0xCFFFFFFF	256MB		-	NS	0xC	0
49	0xD0000000	0xDFFFFFFF	256MB		-	S	0xD	0
50	0xE0000000	0xE003FFFF	256KB	Cortex®-M23 *Private Peripheral Bus* (PPB).	-	Exempt		
51	0xE0040000	0xE0040FFF	4KB	Debug APB expansion interface, *Trace Port Interface Unit* (TPIU).	-			
52	0xE0041000	0xE0041FFF	4KB	Cortex®-M23 *Embedded Trace Macrocell* (ETM).	-			
53	0xE0042000	0xE0042FFF	4KB	Cortex®-M23 *Cross Trigger Interface* (CTI).	-			
54	0xE0043000	0xE00FDFFF	748KB	Reserved.	-			
55	0xE00FE000	0xE00FEFFF	4KB	Debug APB expansion interface.	-			
56	0xE00FF000	0xE00FFFFF	4KB	Cortex®-M23 processor debug ROM.	-			
57	0xE0100000	0xEFFFFFFF	255MB	Peripheral AHB5 expansion Master Interface 1.	-	NS	0xE	0
58	0xF0000000	0xF00FFFFF	1MB	Debug APB expansion interface, system debug region.	-	Exempt		
59	0xF0100000	0xFFFFFFFF	255MB	Peripheral AHB5 expansion master interface 1.	-	S	0xF	0

4.2 Subsystem register descriptions

This section defines the registers in the SSE-123 memory map.

4.2.1 Subsystem information registers block

The SSE-123 information registers provide software information for hardware identification and configuration.

Power domain: **PD_SYS**.

Reset domain: **nHRESET**.

The following table shows the registers in the system information register block.

Table 4-2: Subsystem information register summary

Offset	Name	Type	Reset value	Description	Security
0x000	SYS_VERSION	RO	0x0004_1748	4.2.1.1 SYS_VERSION, System Version register on page 46.	All
0x004	SYS_CONFIG	RO	Depends on subsystem configuration.	4.2.1.2 SYS_CONFIG, System Configuration register on page 47	All
0x008 - 0xFCC	-	-	0x0000_0000	Reserved.	All
0xFD0	PIDR4	RO	0x0000_0004	Peripheral ID 4.	All
0xFD4	PIDR5	-	0x0000_0000	Reserved.	All
0xFD8	PIDR6	-	0x0000_0000	Reserved.	All
0xFDC	PIDR7	-	0x0000_0000	Reserved.	All
0xFE0	PIDR0	RO	0x0000_0058	Peripheral ID 0.	All
0xFE4	PIDR1	RO	0x0000_00B8	Peripheral ID 1.	All
0xFE8	PIDR2	RO	0x0000_000B	Peripheral ID 2.	All
0xFEC	PIDR3	RO	0x0000_0000	Peripheral ID 3.	All
0xFF0	CIDR0	RO	0x0000_000D	Component ID 0.	All
0xFF4	CIDR1	RO	0x0000_00F0	Component ID 1.	All
0xFF8	CIDR2	RO	0x0000_0005	Component ID 2.	All
0xFFC	CIDR3	RO	0x0000_00B1	Component ID 3.	All

4.2.1.1 SYS_VERSION, System Version register

The SYS_VERSION register shows the version of the subsystem.

The SYS_VERSION register characteristics are:

Usage constraints

This register is read-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.1 Subsystem information registers block on page 45.

The following table shows the bit assignments.

Table 4-3: SYS_VERSION register bit assignments

Bits	Name	Function
[31:28]	-	Reserved.
[27:24]	MAJOR_REVISION	Set to `0x0`.
[23:20]	MINOR_REVISION	Set to `0x0`.
[19:12]	DESIGNER_ID	Arm® product with designer code `0x41`
[11:0]	PART_NUMBER	Part number for the subsystem product.

4.2.1.2 SYS_CONFIG, System Configuration register

The SYS_CONFIG register reports the configuration of the subsystem.

The SYS_CONFIG register characteristics are:

Usage constraints

This register is read-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.1 Subsystem information registers block on page 45.

The following table shows the bit assignments.

Table 4-4: SYS_CONFIG register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	-	RAZ/WI	16	`0x00`	Reserved.
[15:14]	DEBUG_MODE	RO	2	The level of debug features that were chosen during the configuration of the subsystem.	Debug functionality included: **0** No debug. **1** No trace. **2** Has trace.
[13]	IO_PORT_ENABLED	RO	1	If the Single I/O Port feature was enabled during the configuration of the subsystem, then the reset value is `0x1`. Otherwise, the reset value is `0x0`.	Processor I/O port enabled.
[12:9]	-	RAZ/WI	4	`0x0`	Reserved.
[8:4]	SRAM_ADDR_WIDTH	RO	5	The SRAM address width that was chosen during the configuration of the subsystem.	SRAM size available. Where $SRAM\_SIZE = 2^{SRAM\_ADDR\_WIDTH}$ Valid range is 13-24 (8KB-16MB).

Bits	Name	Access	Width	Reset value	Description
[3:0]	SRAM_NUM_BANK	RO	4	`0x1`	SRAM number of banks. Fixed in this subsystem and therefore always reads as 1 bank.

4.2.2 Subsystem control registers block

The subsystem control registers block implements registers for power, clocks, resets, and other general system control.

The subsystem control registers block is located in the following domains:

Power domain **PD_AON**
Reset domain **nPORESET**, **nAONRESET**, or **nWARMAONRESET**.

The following table shows the registers in the subsystem control register block. For Write-Access to these registers, only 32-bit writes are supported. Any byte or halfword writes result in the write data being ignored.

Table 4-5: Subsystem control register block

Offset	Name	Access	Reset value	Description	Security
`0x000`	SECDBGSTAT	RO	Defined by input signals	4.2.2.1 SECDBGSTAT, Secure Debug Status register on page 49, Secure debug configuration status register.	SP
`0x004`	SECDBGSET	WO	`0x0000_0000`	4.2.2.2 SECDBGSET, Secure Debug Set register on page 50, Secure debug configuration set register.	SP
`0x008`	SECDBGCLR	WO	`0x0000_0000`	4.2.2.3 SECDBGCLR, Secure Debug Clear register on page 51, Secure debug configuration clear register.	SP
`0x00C`	SCSECCTRL	RW	`0x0000_0000`	4.2.2.4 SCSECCTRL, System Control Security Control register on page 52.	SP
`0x010-0x014`		RAZ/WI	`0x0000_0000`	Reserved.	SP
`0x018`	CLOCK_CTRL	RW	`0x0000_0000`	4.2.2.5 CLOCK_CTRL, Clock Control register on page 52.	SP
`0x01C-0x0FC`		RAZ/WI	`0x0000_0000`	Reserved.	SP
`0x100`	RESET_SYNDROME	RW	`0x0000_0001`	4.2.2.6 RESET_SYNDROME, Reset Syndrome register on page 53.	SP
`0x104`	RESET_MASK	RW	`0x0000_0000`	4.2.2.7 RESET_MASK, Reset Mask register on page 54.	SP
`0x108`	SWRESET	WO	`0x0000_0000`	4.2.2.8 SWRESET, Software Reset register on page 55.	SP
`0x10C`	GRETREG	RW	`0x0000_0000`	4.2.2.9 GRETREG, General Purpose Retention register on page 56.	SP
`0x110`	INITVTOR	RW	Defined by Input Signal	Initial Secure reset vector register for processor, 4.2.2.10 INITVTOR, Initial Secure Vector Table Offset register on page 56	SP
`0x114`	-	RAZ/WI	`0x0000_0000`	Reserved.	SP
`0x118`	CPUWAIT	RW	`0x0000_0000`	Processor boot wait control after reset, 4.2.2.11 CPUWAIT, CPU Wait register on page 57.	SP
`0x11C`	NMI_ENABLE	RO	`0x0001_0001`	NMI enable register, 4.2.2.12 NMI_ENABLE, Non-Maskable Interrupt Enable register on page 57.	SP

Offset	Name	Access	Reset value	Description	Security
`0x120`	WICCTRL	RAZ/WI	`0x0000_0000`	Reserved.	SP
`0x124`	WICBRGCTRL	RW	`0x0000_0000`	WIC bridge control register, 4.2.2.13 WICBRGCTRL, WIC Bridge Control register on page 58.	SP
`0x128-0xFCC`		RAZ/WI	`0x0000_0000`	Reserved.	SP
`0xFD0`	PIDR4	RO	`0x0000_0004`	Peripheral ID 4 register.	SP
`0xFD4`	PIDR5	RO	`0x0000_0000`	Reserved.	SP
`0xFD8`	PIDR6	RO	`0x0000_0000`	Reserved.	SP
`0xFDC`	PIDR7	RO	`0x0000_0000`	Reserved.	SP
`0xFE0`	PIDR0	RO	`0x0000_0054`	Peripheral ID 0 register.	SP
`0xFE4`	PIDR1	RO	`0x0000_00B8`	Peripheral ID 1 register.	SP
`0xFE8`	PIDR2	RO	`0x0000_000B`	Peripheral ID 2 register.	SP
`0xFEC`	PIDR3	RO	`0x0000_0000`	Peripheral ID 3 register.	SP
`0xFF0`	CIDR0	RO	`0x0000_000D`	Component ID 0 register.	SP
`0xFF4`	CIDR1	RO	`0x0000_00F0`	Component ID 1 register.	SP
`0xFF8`	CIDR2	RO	`0x0000_0005`	Component ID 2 register.	SP
`0xFFC`	CIDR3	RO	`0x0000_00B1`	Component ID 3 register.	SP

4.2.2.1 SECDBGSTAT, Secure Debug Status register

The Secure Debug Configuration registers are used to select the source value for the combined Secure Debug Authentication Signals, **DBGEN**, **NIDEN**, **SPIDEN**, and **SPNIDEN**.

For each signal, a selector is provided to select between an internal register value and the value on the boundary of the Subsystem.

Secure software can set or clear each value by setting the associated bit in the 4.2.2.2 SECDBGSET, Secure Debug Set register on page 50 or in the 4.2.2.3 SECDBGCLR, Secure Debug Clear register on page 51, respectively. Secure software can read the system-wide value by reading the associated SECDBGSTAT register bit.

For example, the DBGEN_SEL register selects the source of the DBGEN value that is used in the system, where:

- If DBGEN_SEL is LOW, the **DBGENIN** input signal is used to define the system-wide DBGEN value.
- If DBGEN_SEL is HIGH, the internal register value for DBGEN is used to define the system-wide DBGEN value.

To set the DBGEN or DBGEN_SEL register values HIGH, write to the SECDBGSET register with DBGEN_SET or DBGEN_SEL_SET set to HIGH, respectively.

To set the DBGEN and DBGEN_SEL register values to LOW, write to the SECDBGCLR register with DBGEN_CLR or DBGEN_SEL_CLR set to HIGH, respectively.

To read the value of the selected DBGEN, read the SECDBGSTAT register for the DBGEN_SEL_STAT value.

The selected DGBEN register value is also made available to external expansion logic through the **DBGEN** output signal of the Subsystem.

Top-level Static Configuration signals, **DBGEN_SEL_DIS**, **NIDEN_SEL_DIS**, **SPIDEN_SEL_DIS**, and **SPNIDEN_SEL_DIS**, are provided to disable each of the selectors. Disabling the selectors forces the corresponding registers DBGEN_SEL_STATUS, NIDEN_SEL_STATUS, SPIDEN_SEL_STATUS, and SPNIDEN_SEL_STATUS to zero, forcing each respective input to use its external value.

This can be used to disable the ability for Secure firmware to modify or override the Debug Authentication value.

The reset domain is **nAONRESET**.

The SECDBGSTAT register characteristics are:

Usage constraints

This register is read-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-6: SECDBGSTAT register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:8]	-	RAZ/ WI	24	`0x000000`	Reserved.
[7]	SPNIDEN_SEL_STATUS	RO	1	`0x0`	Active-HIGH Secure privilege non-invasive debug enable selector value.
[6]	SPNIDEN_STATUS	RO	1	Input signal **SPNIDENIN**	Active-HIGH Secure privilege non-invasive debug enable value.
[5]	SPIDEN_SEL_STATUS	RO	1	`0x0`	Active-HIGH Secure privilege invasive debug enable selector value.
[4]	SPIDEN_STATUS	RO	1	Input signal **SPIDENIN**	Active-HIGH Secure privilege invasive debug enable value.
[3]	NIDEN_SEL_STATUS	RO	1	`0x0`	Active-HIGH non-invasive debug enable selector value.
[2]	NIDEN_STATUS	RO	1	Input signal **NIDENIN**	Active-HIGH non-invasive debug enable value.
[1]	DBGEN_SEL_STATUS	RO	1	`0x0`	Active-HIGH debug enable selector value.
[0]	DBGEN_STATUS	RO	1	Input signal **DBGENIN**	Active-HIGH debug enable value.

4.2.2.2 SECDBGSET, Secure Debug Set register

The Secure Debug Configuration registers are used to select the source value for the combined Secure Debug Authentication Signals, **DBGEN**, **NIDEN**, **SPIDEN**, and **SPNIDEN**.

See 4.2.2.1 SECDBGSTAT, Secure Debug Status register on page 49 for a description of this register.

The SECDBGSET register characteristics are:

Usage constraints

This register is write-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-7: SECDBGSET register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:8]	-	RAZ/WI	24	`0x000000`	Reserved.
[7]	SPNIDEN_SEL_SET	WO	1	`0x0`	Active-HIGH Secure privilege non-invasive debug enable selector value.
[6]	SPNIDEN_SET	WO	1	`0x0`	Active-HIGH Secure privilege non-invasive debug enable value.
[5]	SPIDEN_SEL_SET	WO	1	`0x0`	Active-HIGH Secure privilege invasive debug enable selector value.
[4]	SPIDEN_SET	WO	1	`0x0`	Active-HIGH Secure privilege invasive debug enable value.
[3]	NIDEN_SEL_SET	WO	1	`0x0`	Active-HIGH non-invasive debug enable selector value.
[2]	NIDEN_SET	WO	1	`0x0`	Active-HIGH non-invasive debug enable value.
[1]	DBGEN_SEL_SET	WO	1	`0x0`	Active-HIGH debug enable selector value.
[0]	DBGEN_SET	WO	1	`0x0`	Active-HIGH debug enable value.

4.2.2.3 SECDBGCLR, Secure Debug Clear register

The Secure Debug Configuration registers are used to select the source value for the combined Secure Debug Authentication Signals, **DBGEN**, **NIDEN**, **SPIDEN**, and **SPNIDEN**.

See 4.2.2.1 SECDBGSTAT, Secure Debug Status register on page 49 for a description of this register.

The SECDBGCLR register characteristics are:

Usage constraints

This register is write-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-8: SECDBGCLR register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:8]	-	RAZ/WI	24	`0x000000`	Reserved.
[7]	SPNIDEN_SEL_CLR	WO	1	`0x0`	Active-HIGH Secure privilege non-invasive debug enable selector value.
[6]	SPNIDEN_CLR	WO	1	`0x0`	Active-HIGH Secure privilege non-invasive debug enable value.
[5]	SPIDEN_SEL_CLR	WO	1	`0x0`	Active-HIGH Secure privilege invasive debug enable selector value.
[4]	SPIDEN_CLR	WO	1	`0x0`	Active-HIGH Secure privilege invasive debug enable value.
[3]	NIDEN_SEL_CLR	WO	1	`0x0`	Active-HIGH non-invasive debug enable selector value.
[2]	NIDEN_CLR	WO	1	`0x0`	Active-HIGH non-invasive debug enable value.
[1]	DBGEN_SEL_CLR	WO	1	`0x0`	Active-HIGH debug enable selector value.
[0]	DBGEN_CLR	WO	1	`0x0`	Active-HIGH debug enable value.

4.2.2.4 SCSECCTRL, System Control Security Control register

The SCSECCTRL register provides register bits to set the Secure configuration lock of this register block. It is located in the **nPORESET** reset domain.

The SCSECCTRL register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-9: SCSECCTRL register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:3]	-	RAZ/WI	29	`0x00000000`	Reserved.
[2]	SCSECCFGLOCK	RW. Write one to set. Write zero ignored.	1	`0x0`	Active-HIGH control to disable writes to security-related control registers in this register block. When set to HIGH, it can no longer be cleared to zero except by using a Power On Reset. When set to HIGH, Write-Access to SECDBGSET, SECDBGCLR, and INITVTOR are ignored.
[1:0]	-	RAZ/WI	2	`0x0`	Reserved.

4.2.2.5 CLOCK_CTRL, Clock Control register

The CLOCK_CTRL register enables software to control dynamic clock gating and dynamic clock switching behavior. By default, these power-saving features are not enabled. Software must set these features accordingly during boot to use the power-saving benefits.

The CLOCK_CTRL register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-10: CLOCK_CTRL register bit assignments

Bits	Name	Access	Width	Reset Value	Description
[31:18]	-	RAZ/WI	14	`0x0000`	Reserved.
[17:16]	FCLK_DCS_ENABLE	RW	2	`0x0`	Defines when dynamic clock switching is enabled for the **FCLK** domain: **`2'b00`** Dynamic clock switching is disabled. **`2'b01`** Dynamic clock switching is enabled when the Cortex®-M23 processor core enters DEEPSLEEP. **`2'b10`** Dynamic clock switching is enabled when the subsystem enters the CSS.SLEEP power state. **`2'b11`** Reserved. Invalid state, dynamic clock switching is disabled.
[15:2]	-	RAZ/WI	14	`0x0000`	Reserved.
[1]	DCLK_FORCE	RW	1	`0x1`	Set this bit as follows: **`0`** Dynamic clock gating is enabled for the **DCLK** domain. **`1`** Dynamic clock gating is disabled for the **DCLK** domain.
[0]	HCLK_FORCE	RW	1	`0x1`	Set this bit as follows: **`0`** Dynamic clock gating is enabled for the **HCLK** domain. **`1`** Dynamic clock gating is disabled for the **HCLK** domain.

4.2.2.6 RESET_SYNDROME, Reset Syndrome register

The RESET_SYNDROME register stores the reason for the last reset event and either software or **nPORESET** clears this register. Writing zero clears all fields. Write one is ignored and maintains the previous value for that bit. The RESET_SYNDROME register is in the **nPORESET** reset domain.

Note

CPULOCKUP does not generate reset, but when HIGH, it indicates that the processor has locked-up. Locking-up could be a precursor to another reset event, for example, a watchdog timer reset request.

The RESET_SYNDROME register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-11: RESET_SYNDROME register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:10]	-	RAZ/WI	22	`0x000000`	Reserved.
[9]	SWRESETREQ	RW. Write zero to clear.	1	`0x0`	Software reset request.
[8]	HWRESETREQ		1	`0x0`	External hardware reset request.
[7]	-	RAZ/WI	1	`0x0`	Reserved.
[6]	CPULOCKUP	RW. Write zero to clear.	1	`0x0`	Processor lock-up status.
[5]	-	RAZ/WI	1	`0x0`	Reserved.
[4]	SYSRESETREQ	RW. Write zero to clear.	1	`0x0`	Processor System Reset Request,AIRCR.SYSRESETREQ.
[3]	-	RAZ/WI	1	`0x0`	Reserved.
[2]	SWD	RW. Write zero to clear.	1	`0x0`	Secure watchdog.
[1]	NSWD		1	`0x0`	Non-secure watchdog.
[0]	PoR		1	`0x1`	Power-On.

4.2.2.7 RESET_MASK, Reset Mask register

The RESET_MASK register enables software to control the reset requests that can cause RESET. Set each bit HIGH to enable each source. The RESET_MASK register is in the **nWARMAONRESET** reset domain.

Note

Each mask bit, if cleared, prevents the reset source being used to generate the reset. Clearing mask bits also prevents the associated RESET_SYNDROME register bit from recording the event.

The RESET_MASK register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-12: RESET_MASK register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:5]	-	RAZ/WI	27	`0x00000000`	Reserved.
[4]	SYSRESETREQ_EN	RW	1	`0x0`	Enable AIRCR.SYSRESETREQ reset.
[3:2]	-	RAZ/WI	2	`0x0`	Reserved.
[1]	NSWD_EN	RW	1	`0x0`	Enable Non-secure watchdog reset.
[0]	-	RAZ/WI	1	`0x0`	Reserved.

4.2.2.8 SWRESET, Software Reset register

The SWRESET register enables software to request a system reset.

The SWRESET register characteristics are:

Usage constraints

This register is write-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-13: SWRESET register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:10]	-	RAZ/WI	22	0x000000	Reserved.
[9]	SWRESETREQ	WO	1	0x0	Software reset request. Set HIGH to request a system reset that is equivalent to a watchdog or Cold reset.
[8:0]	-	RAZ/WI	9	0x0	Reserved.

4.2.2.9 GRETREG, General Purpose Retention register

The GRETREG register provides 16 bits of retention register for general storage, especially during power down of the rest of the system. The GRETREG register is in the **nAONRESET** reset domain.

The GRETREG register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-14: GRETREG register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	-	RAZ/WI	16	0x000000	Reserved.
[15:0]	GRETREG	RW	16	0x0000	General purpose retention register.

4.2.2.10 INITVTOR, Initial Secure Vector Table Offset register

The INITVTOR register defines the processor initial Secure vector table offset, VTOR_S.TBLOFF[31:8], out of reset. The INITVTOR register is in the **nWARMAONRESET** reset domain.

The INITVTOR register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-15: INITVTOR register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:8]	INITVTOR	RW	24	**INITVTOR_RST[23:0]** input signal.	Default Secure vector table offset at reset for the processor.
[7:0]	-	RAZ/WI	8	-	Reserved.

4.2.2.11 CPUWAIT, CPU Wait register

The CPUWAIT register provides controls to force the processor to wait after reset rather than boot immediately. This forced waiting enables another entity in the expansion system or the debugger to access the system before the processor booting. The CPUWAIT register is in the **nPORESET** reset domain.

The CPUWAIT register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-16: CPUWAIT register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:1]	-	RAZ/WI	31	-	Reserved.
[0]	CPUWAIT	RW	1	**CPUWAIT_RST** input signal	Processor waits at boot: **0** Boot normally. **1** Wait at boot. This register can be cleared to zero after reset when the **CPUWAIT_CLR** input signal is set HIGH, enabling the processor to boot after some setup has been completed external to the subsystem. See A.17 System control signals on page 105.

4.2.2.12 NMI_ENABLE, Non-Maskable Interrupt Enable register

The NMI_ENABLE register provides controls to enable or disable internally or externally generated *Non-Maskable Interrupt* (NMI) sources from generating an NMI interrupt on specific processor cores. Because this subsystem has only a single processor core, these registers are read-only and set to always enable the non-maskable interrupts. These registers are provided only for software compatibility with other IoT subsystems. The NMI_ENABLE register is in the **nAONRESET** reset domain.

The NMI_ENABLE register characteristics are:

Usage constraints

This register is read-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-17: NMI_ENABLE register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:17]	-	RAZ/WI	15	-	Reserved.
[16]	EXTNMI_ENABLE	RO	1	1	Externally-sourced NMI enable. This bit determines whether the top-level pin, **EXPNMI**, can raise an NMI interrupt on the processor: **HIGH** Permitted. **LOW** Masked and not permitted.
[15:1]	-	RAZ/WI	15	-	Reserved.
[0]	INTNMI_ENABLE	RO	1	1	Internally-sourced NMI enable. This bit determines whether the subsystem internally generated NMI interrupt sources can raise an NMI interrupt on the processor: **HIGH** Permitted. **LOW** Masked and not permitted.

4.2.2.13 WICBRGCTRL, WIC Bridge Control register

The WICBRGCTRL register permits software to enable the *Wakeup Interrupt Controller* (WIC) bridge and read the status of the bridge. The WICBRGCTRL register is in the **nAONRESET** reset domain.

The WICBRGCTRL register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.2 Subsystem control registers block on page 48.

The following table shows the bit assignments.

Table 4-18: WICBRGCTRL register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:9]	-	RAZ/WI	23	-	Reserved.
[8]	WICBRGEN_CLR	WO	1	`0x0`	WIC bridge enable clear. Writing '1' to this bit clears **WICBRGEN_STATUS** to LOW. This field always reads as '0'.
[7:5]	-	RAZ/WI	3	-	Reserved.
[4]	WICBRGEN_SET	WO	1	`0x0`	WIC bridge enable set. Writing '1' to this bit sets **WICBRGEN_STATUS** to HIGH. This field always reads as '0'.
[3:1]	-	RAZ/WI	3	-	Reserved.
[0]	WICBRGEN_STATUS	RO	1	`0x0`	WIC bridge enable status. When set HIGH using the WICBRGEN_SET register, enables the WIC bridge to isolate communication between the WIC and NVIC. For more information, see 3.3.7 Wakeup Interrupt Controller (WIC) and WIC-bridge on page 32.

4.2.3 Secure privilege control registers block

The Secure privilege control register block implements program-visible states that enable software to control security gating units within the design. These registers are Secure privileged access only and support 32-bit R/W accesses.

Power domain: **PD_SYS**.

Reset domain: **nHRESET**.

The following table shows the registers in the Secure privilege control registers block. For Write-Access to these registers, only 32-bit writes are supported. Any byte and halfword writes result in the write data being ignored.

Table 4-19: Secure privilege control registers block

Offset	Name	Access	Reset value	Description	Security
0x000	SPCSECCTRL	RW	0x0000_0000	4.2.3.1 SPCSECCTRL, Secure Privilege Controller Secure Configuration Control register on page 62.	SP
0x004	BUSWAIT	RW	0x0000_0000	4.2.3.2 BUSWAIT, Bus Access Wait register on page 62.	SP
0x004-0x010		RAZ/WI	0x0000_0000	Reserved.	SP
0x010	SECRESPCFG	RW	0x0000_0000	4.2.3.3 SECRESPCFG, Security Violation Response Configuration register on page 63.	SP
0x014	NSCCFG	RW	0x0000_0000	4.2.3.4 NSCCFG, Non-secure Callable Configuration register on page 64.	SP
0x018	-	RAZ/WI	0x0000_0000	Reserved.	SP
0x01C	SECMPCINTSTATUS	RO	0x0000_0000	4.2.3.5 SECMPCINTSTATUS, Secure MPC Interrupt Status register on page 65.	SP
0x020	SECPPCINTSTAT	RO	0x0000_0000	4.2.3.6 SECPPCINTSTAT, Secure PPC Interrupt Status register on page 66.	SP
0x024	SECPPCINTCLR	WO	0x0000_0000	4.2.3.7 SECPPCINTCLR, Secure PPC Interrupt Clear register on page 66.	SP
0x028	SECPPCINTEN	RW	0x0000_0000	4.2.3.8 SECPPCINTEN, Secure PPC Interrupt Enable register on page 67.	SP
0x02C-0x06C		RAZ/WI	0x0000_0000	Reserved.	SP
0x030	SECMSCINTSTAT	RO	0x0000_0000	4.2.3.9 SECMSCINTSTAT, Secure MSC Interrupt Status register on page 68.	SP
0x034	SECMSCINTCLR	WO	0x0000_0000	4.2.3.10 SECMSCINTCLR, Secure MSC Interrupt Clear register on page 69.	SP
0x038	SECMSCINTEN	RW	0x0000_0000	4.2.3.11 SECMSCINTEN, Secure MSC Interrupt Enable register on page 70.	SP
0x03C	-	RAZ/WI	0x0000_0000	Reserved.	SP
0x040	BRGINTSTAT	RO	0x0000_0000	4.2.3.12 BRGINTSTAT, Bridge Interrupt Status register on page 71.	SP
0x044	BRGINTCLR	WO	0x0000_0000	4.2.3.13 BRGINTCLR, Bridge Interrupt Clear register on page 72.	SP
0x048	BRGINTEN	RW	0x0000_0000	4.2.3.14 BRGINTEN, Bridge Interrupt Enable register on page 73.	SP
0x04C	-	RAZ/WI	0x0000_0000	Reserved.	SP
0x050-0x05C		RAZ/WI	0x0000_0000	Reserved for future Non-secure access AHB slave peripheral protection controller, AHBNSPPC0-3.	SP
0x060	AHBNSPPCEXP	RW	0x0000_0000	4.2.3.15 AHBNSPPCEXP, AHB Non-secure Access PPC Expansion register on page 74.	SP
0x064-0x06C		RAZ/WI	0x0000_0000	Reserved, AHBNSPPCEXP1-3.	SP
0x070	APBNSPPC	RW	0x0000_0000	Non-secure access APB slave peripheral protection control.	SP
0x074-0x07C		RAZ/WI	0x0000_0000	Reserved for future Non-secure access APB slave peripheral protection controller, APBNSPPC1-3.	SP

Offset	Name	Access	Reset value	Description	Security
0x080	APBNSPPCEXP0	RW	0x0000_0000	4.2.3.17 APBNSPPCEXP0, APBNSPPCEXP1, APBNSPPCEXP2, and APBNSPPCEXP3, APB Non-secure Access PPC Expansion registers on page 75.	SP
0x084	APBNSPPCEXP1	RW	0x0000_0000	4.2.3.17 APBNSPPCEXP0, APBNSPPCEXP1, APBNSPPCEXP2, and APBNSPPCEXP3, APB Non-secure Access PPC Expansion registers on page 75.	SP
0x088	APBNSPPCEXP2	RW	0x0000_0000	4.2.3.17 APBNSPPCEXP0, APBNSPPCEXP1, APBNSPPCEXP2, and APBNSPPCEXP3, APB Non-secure Access PPC Expansion registers on page 75.	SP
0x08C	APBNSPPCEXP3	RW	0x0000_0000	4.2.3.17 APBNSPPCEXP0, APBNSPPCEXP1, APBNSPPCEXP2, and APBNSPPCEXP3, APB Non-secure Access PPC Expansion registers on page 75.	SP
0x090-0x09C		RAZ/WI	0x0000_0000	Reserved for future Secure privileged access AHB slave peripheral protection controller, AHBSPPPC0-3.	SP
0x0A0	AHBSPPPCEXP	RW	0x0000_0000	4.2.3.18 AHBSPPPCEXP, AHB Secure Privilege Access PPC Expansion register on page 76.	SP
0x0A4-0x0AC		RAZ/WI	0x0000_0000	Reserved, AHPSPPPCEXP1-3.	SP
0x0B0	APBSPPPC	RW	0x0000_0000	4.2.3.19 APBSPPPC, APB Secure Privilege Access PPC register on page 77.	SP
0x0b4-0x0BC		RAZ/WI	0x0000_0000	Reserved for future Secure unprivileged access APB slave peripheral protection controller, APBSPPC1-3.	SP
0x0C0	APBSPPPCEXP0	RW	0x0000_0000	4.2.3.20 APBSPPPCEXP0, APBSPPPCEXP1, APBSPPPCEXP2, and APBSPPPCEXP3, APB Secure Privilege Access PPC Expansion registers on page 78.	SP
0x0C4	APBSPPPCEXP1	RW	0x0000_0000	4.2.3.20 APBSPPPCEXP0, APBSPPPCEXP1, APBSPPPCEXP2, and APBSPPPCEXP3, APB Secure Privilege Access PPC Expansion registers on page 78.	SP
0x0C8	APBSPPPCEXP2	RW	0x0000_0000	4.2.3.20 APBSPPPCEXP0, APBSPPPCEXP1, APBSPPPCEXP2, and APBSPPPCEXP3, APB Secure Privilege Access PPC Expansion registers on page 78.	SP
0x0CC	APBSPPPCEXP3	RW	0x0000_0000	4.2.3.20 APBSPPPCEXP0, APBSPPPCEXP1, APBSPPPCEXP2, and APBSPPPCEXP3, APB Secure Privilege Access PPC Expansion registers on page 78.	SP
0x0D0	NSMSCEXP	RW	0x0000_0000	4.2.3.21 NSMSCEXP, Non-secure Access MSC Expansion register on page 79.	SP
0x0D4-0xFCC		RAZ/WI	0x0000_0000	Reserved.	SP
0xFD0	PIDR4	RO	0x0000_0004	Peripheral ID 4.	SP
0xFD4	PIDR5	RO	0x0000_0000	Reserved.	SP
0xFD8	PIDR6	RO	0x0000_0000	Reserved.	SP
0xFDC	PIDR7	RO	0x0000_0000	Reserved.	SP
0xFE0	PIDR0	RO	0x0000_0052	Peripheral ID 0.	SP
0xFE4	PIDR1	RO	0x0000_00B8	Peripheral ID 1.	SP
0xFE8	PIDR2	RO	0x0000_000B	Peripheral ID 2.	SP
0xFEC	PIDR3	RO	0x0000_0000	Peripheral ID 3.	SP
0xFF0	CIDR0	RO	0x0000_000D	Component ID 0.	SP

Offset	Name	Access	Reset value	Description	Security
`0xFF4`	CIDR1	RO	`0x0000_00F0`	Component ID 1.	SP
`0xFF8`	CIDR2	RO	`0x0000_0005`	Component ID 2.	SP
`0xFFC`	CIDR3	RO	`0x0000_00B1`	Component ID 3.	SP

4.2.3.1 SPCSECCTRL, Secure Privilege Controller Secure Configuration Control register

The SPCSECCTRL register implements the security lock register.

The SPCSECCTRL register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-20: SPCSECCTRL register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:1]	-	RAZ/WI	31	-	Reserved.
[0]	SPCSECCFGLOCK	RW	1	`0x0`	Write '1' to set. Cleared by reset only. Disables writes to all security-related registers in the Secure privilege control registers block. When SPCSECCFGLOCK is set to HIGH, the following registers can no longer be modified until reset: • NSCCFG. • APBNSPPC. • APBSPPPC. • AHBNSPPCEXP. • AHBSPPPCEXP. • APBNSPPCEXP. • APBSPPPCEXP. • NSMSCEXP.

4.2.3.2 BUSWAIT, Bus Access Wait register

The BUSWAIT register enables software to indicate when the configuration of the MPCs or other Security registers is complete. This indication enables components outside the subsystem to gate access to the subsystem, for example, an AHB5 *Access Control Gate* (ACG) until the security settings have been applied.

The BUSWAIT register characteristics are:

Usage constraints

Bit[16] RO, bit[0] RW.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-21: BUSWAIT register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:17]	-	RAZ/WI	15	-	Reserved.
[16]	ACC_WAITN_STATUS	RO	1	`0x0`	This status register indicates the status of any gating units that block bus access to the subsystem: **0** Block access. **1** Allow access. The **ACCWAITN_STAT** input signal sets the status of this register.
[15:1]	-	RAZ/WI	15	-	Reserved.
[0]	ACC_WAITN	RW	1	Defined by the `ACC_WAITN_RST` parameter. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.	Request gating units to block bus access to subsystem: **0** Block access. **1** Allow access. This register drives the **ACCWAITN** output signal.

4.2.3.3 SECRESPCFG, Security Violation Response Configuration register

The SECRESPCFG register is used to define a slave response to an access that causes a security violation in the subsystem.

The SECRESPCFG register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-22: SECRESPCFG register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:1]	-	RAZ/WI	31	-	Reserved.
[0]	SECRESPCFG	RW	1	`0x0`	This field configures the slave response if there is a security violation: **0** Read-As-Zero, Write Ignore. **1** Bus error. **Note:** Some slaves, for example, the MPCs, provide their own control registers to configure their own response. These slaves do not depend on this control bit.

4.2.3.4 NSCCFG, Non-secure Callable Configuration register

The NSCCFG register enables software to define whether the Secure code region `0x1000_0000` to `0x1FFF_FFFF`, and the Secure SRAM region `0x3000_0000` to `0x3FFF_FFFF` are Non-secure callable regions of memory.

The NSCCFG register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-23: NSCCFG register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:2]	-	RAZ/WI	30	-	Reserved.

Bits	Name	Access	Width	Reset value	Description
[1]	RAMNSC	RW	1	`0x0`	Configures whether the RAM region, `0x3000_0000` to `0x3FFF_FFFF` is Non-secure callable: **0** Not Non-secure callable. **1** Non-secure callable.
[0]	CODENSC	RW	1	`0x0`	Configures whether the CODE region, `0x1000_0000` to `0x1FFF_FFFF`, is Non-secure callable: **0** Not Non-secure callable. **1** Non-secure callable.

4.2.3.5 SECMPCINTSTATUS, Secure MPC Interrupt Status register

The interrupt signals from the *Memory Protection Controller* (MPC), both within the subsystem and in the expansion logic, are merged and sent to the processor NVIC on a single interrupt signal. The Secure MPC interrupt status register enables Secure software to check which MPC is causing the interrupt. When the source of the interrupt is identified, you must use the MPC register interface to clear the interrupt.

The SECMPCINTSTATUS register characteristics are:

Usage constraints

This register is read-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-24: SECMPCINTSTATUS register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	S_MPCEXP_STATUS	RO	16	`0x0000`	Interrupt status for expansion memory protection controller. Each bit 'n' shows the status of the **SMPCEXP_STAT[n]** input signal. The `MPCEXP_DIS` parameter defines whether each bit within this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If MPCEXP_DIS[n] = `1'b1`, then **SMPCEXP_STAT[n]** is disabled and always reads as zeros.
[15:1]	-	RAZ/WI	15	-	Reserved.
[0]	S_MPCSRAM_STATUS	RO	1	`0x0`	Interrupt status for memory protection controller of SRAM.

4.2.3.6 SECPPCINTSTAT, Secure PPC Interrupt Status register

When access violations occur on any *Peripheral Protection Controller* (PPC), a level interrupt is raised using a combined interrupt to the processor *Nested Vector Interrupt Controller* (NVIC). The Secure PPC interrupt status, clear, and enable registers permit software to determine the source, clear, or mask the PPC interrupt.

The SECPPCINTSTAT register characteristics are:

Usage constraints

This register is read-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-25: SECPPCINTSTAT register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:24]	-	RAZ/WI	8	-	Reserved.
[23:20]	S_AHBPPCEXP_STATUS	RO	4	`0x0`	Interrupt status of the expansion peripheral protection controller for AHB slaves. Each bit 'n' shows the status of the **SAHBPPCEXP_STAT[n]** input signal.
[19:16]	-	RAZ/WI	4	-	Reserved.
[15:8]	-	RAZ/WI	8	-	Reserved.
[7:4]	S_APBPPCEXP_STATUS	RO	4	`0x0`	Interrupt status of expansion peripheral protection controller for APB slaves. Each bit 'n' shows the status of the **SAPBPPCEXP_STAT[n]** input signal.
[3:1]	-	RAZ/WI	3	-	Reserved.
[0]	S_APBPPCPERIP_STATUS	RO	1	`0x0`	Interrupt status of subsystem peripheral protection controller.

4.2.3.7 SECPPCINTCLR, Secure PPC Interrupt Clear register

When access violations occur on any *Peripheral Protection Controller* (PPC), a level interrupt is raised using a combined interrupt to the processor NVIC. The Secure PPC interrupt status, clear, and enable registers permit software to determine the source, clear, or mask the PPC interrupt.

The SECPPCINTCLR register characteristics are:

Usage constraints

This register is write-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-26: SECPPCINTCLR register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:24]	-	RAZ/WI	8	-	Reserved.
[23:20]	S_AHBPPCEXP_CLR	WO	4	`0x0`	Interrupt clear of the expansion peripheral protection controller for AHB slaves. Write '1' to clear. Automatically returns to '0' when the associated interrupt status clears. Each bit 'n' drives the **SAHBPPCEXP_CLR[n]** output signal.
[19:16]	-	RAZ/WI	4	-	Reserved.
[15:8]	-	RAZ/WI	8	-	Reserved.
[7:4]	S_APBPPCEXP_CLR	WO	4	`0x0`	Interrupt clear of the expansion peripheral protection controller for APB slaves. Write '1' to clear. Automatically returns to '0' when the associated interrupt status clears. Each bit 'n' drives the **SAPBPPCEXP_CLR[n]** input signal.
[3:1]	-	RAZ/WI	3	-	Reserved.
[0]	S_APBPPCPERIP_CLR	WO	1	`0x0`	Interrupt clear of the subsystem peripheral protection controller. Write '1' to clear. Automatically returns to '0' when the associated interrupt status clears.

4.2.3.8 SECPPCINTEN, Secure PPC Interrupt Enable register

When access violations occur on any *Peripheral Protection Controller* (PPC), a level interrupt is raised using a combined interrupt to the processor *Nested Vector Interrupt Controller* (NVIC). The Secure

PPC interrupt status, clear, and enable registers permit software to determine the source, clear, or mask the PPC interrupt.

The SECPPCINTEN register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-27: SECPPCINTEN register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:24]	-	RAZ/WI	8	-	Reserved.
[23:20]	S_AHBPPCEXP_EN	RW	4	`0x0`	Interrupt enable of the expansion peripheral protection controller for AHB slaves. **0** Write '0' to mask this interrupt source. **1** Write '1' to enable this interrupt source. Each bit 'n' enables or disables **SAHBPPCEXP_STAT[n]**.
[19:16]	-	RAZ/WI	4	-	Reserved.
[15:8]	-	RAZ/WI	8	-	Reserved.
[7:4]	S_APBPPCEXP_EN	RW	4	`0x0`	Interrupt enable of the expansion peripheral protection controller for APB slaves. **0** Write '0' to mask this interrupt source. **1** Write '1' to enable this interrupt source. Each bit 'n' enables or disables an interrupt from **SAPBPPCEXP_STAT[n]**.
[3:1]	-	RAZ/WI	3	-	Reserved.
[0]	S_APBPPCPERIP_EN	RW	1	`0x0`	Interrupt enable of the subsystem peripheral protection controller. **0** Write '0' to mask this interrupt source. **1** Write '1' to enable this interrupt source.

4.2.3.9 SECMSCINTSTAT, Secure MSC Interrupt Status register

When security violation occurs at any *Master Security Controller* (MSC) from outside the subsystem, in the expansion logic, an interrupt is raised using a combined interrupt to the processor *Nested*

Vector Interrupt Controller (NVIC). The Secure MSC interrupt status, clear, and enable registers permit software to determine the source of the interrupt, clear the interrupt, and enable or disable (mask) the interrupt.

The SECMSCINTSTAT register characteristics are:

Usage constraints

This register is read-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-28: SECMSCINTSTAT register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	S_MSCEXP_STATUS	RO	16	`0x0000`	Interrupt status for expansion MSC. Each bit 'n' shows the status of the **SMSCEXP_STAT[n]** input signal. The `MSCEXP_DIS` parameter defines whether each bit within this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If **MSCEXP_DIS[n]** = `1'b1`, then **SMSCEXP_STAT[n]** is disabled and always reads as zero.
[15:0]	-	RAZ/WI	16	-	Reserved.

4.2.3.10 SECMSCINTCLR, Secure MSC Interrupt Clear register

When security violation occurs at any *Master Security Controller* (MSC) from outside the subsystem, in the expansion logic, an interrupt is raised using a combined interrupt to the processor *Nested Vector Interrupt Controller* (NVIC). The Secure MSC interrupt status, clear, and enable registers permit software to determine the source of the interrupt, clear the interrupt, and enable or disable (mask) the interrupt.

The SECMSCINTCLR register characteristics are:

Usage constraints

This register is write-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-29: SECMSCINTCLR register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	S_MSCEXP_CLR	WO	16	`0x0000`	Interrupt clear for expansion MSC. Each bit 'n' drives the **SMSCEXP_CLR[n]** output signal. Write ‘1’ to clear. Automatically returns to ‘0’ when the associated interrupt status clears. The `MSCEXP_DIS` parameter defines whether each bit in this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If **MSCEXP_DIS[n]** = `1'b1`, then **SMSCEXP_CLR[n]** is disabled and any writes to it are ignored.
[15:0]	-	RAZ/WI	16	-	Reserved.

4.2.3.11 SECMSCINTEN, Secure MSC Interrupt Enable register

When security violation occurs at any *Master Security Controller* (MSC) from outside the subsystem, in the expansion logic, an interrupt is raised using a combined interrupt to the processor *Nested Vector Interrupt Controller* (NVIC). The Secure MSC interrupt status, clear, and enable registers permit software to determine the source of the interrupt, clear the interrupt, and enable or disable (mask) the interrupt.

The SECMSCINTEN register characteristics are:

Usage constraints

This register is read-write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-30: SECMSCINTEN register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	S_MSCEXP_EN	RW	16	`0x0000`	Interrupt enable for expansion MSC. Each bit 'n' enables or disables the **SMSCEXP_STAT[n]** input interrupt signal. The `MSCEXP_DIS` parameter defines whether each bit within this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If **MSCEXP_DIS[n]** = `1'b1`, then **SMSCEXP_EN[n]** is disabled and any writes to it are ignored.
[15:0]	-	RAZ/ WI	16	-	Reserved.

4.2.3.12 BRGINTSTAT, Bridge Interrupt Status register

The expansion logic might contain AHB bridges that are necessary to handle clock domain crossing. To improve system performance, some of these bridges can buffer write data, and complete a write access on their slave interfaces before any potential error response is received for the Write-Access on their master interfaces. When this occurs, these bridges can raise a combined interrupt to the processor *Nested Vector Interrupt Controller* (NVIC). The bridge buffer error interrupt status, clear, and enable registers permit software to determine source of the interrupt, clear the interrupt, and enable or disable (mask), the interrupt.

The BRGINTSTAT register characteristics are:

Usage constraints

This register is read-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-31: BRGINTSTAT register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	BRGEXP_STATUS	RO	16	`0x0000`	Interrupt status for expansion bridge buffer error. Each bit 'n' shows the status of the **BRGEXP_STAT[n]** input signal. The `BRGEXP_DIS` parameter defines whether each bit within this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If BRGEXP_DIS[n] = `1'b1`, then **BRGEXP_STAT[n]** is disabled and always reads as zero.
[15:0]	-	RAZ/WI	16	-	Reserved.

4.2.3.13 BRGINTCLR, Bridge Interrupt Clear register

The expansion logic might contain AHB bridges that are necessary to handle clock domain crossing. To improve system performance, some of these bridges can buffer write data, and complete a Write-Access on their slave interfaces before any potential error response is received for the Write-Access on their master interfaces. When this occurs, these bridges can raise a combined interrupt to the processor *Nested Vector Interrupt Controller* (NVIC). The bridge buffer error interrupt status, clear, and enable registers permit software to determine the source of the interrupt, clear the interrupt, and enable or disable (mask), the interrupt.

The BRGINTCLR register characteristics are:

Usage constraints

This register is write-only.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-32: BRGINTCLR register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	BRGEXP_CLR	WO	16	`0x0000`	Interrupt clear for expansion bridge buffer error. Each bit 'n' drives the **BRGEXP_CLR[n]** output signal. Write '1' to clear. Automatically returns to '0' when the associated interrupt status clears. The `BRGEXP_DIS` parameter defines whether each bit within this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If BRGEXP_DIS[n] = `1'b1`, then BRGEXP_CLR[n] is disabled and any writes to it are ignored.
[15:0]	-	RAZ/WI	16	-	Reserved.

4.2.3.14 BRGINTEN, Bridge Interrupt Enable register

The expansion logic might contain AHB bridges that are necessary to handle clock domain crossing. To improve system performance, some of these bridges can buffer write data, and complete a Write-Access on their slave interfaces before any potential error response is received for the Write-Access on their master interfaces. When this occurs, these bridges can raise a combined interrupt to the processor *Nested Vector Interrupt Controller* (NVIC). The bridge buffer error interrupt status, clear, and enable registers permit software to determine source of the interrupt, clear the interrupt, and enable or disable (mask), the interrupt.

The BRGINTEN register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-33: BRGINTEN register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	BRGEXP_EN	RW	16	`0x0000`	Interrupt enable for expansion bridge buffer error. Each bit 'n' enables or disables the **BRGEXP_STAT[n]** input interrupt signal. The `BRGEXP_DIS` parameter defines whether each bit in this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If BRGEXP_DIS[n] = `1'b1`, then BRGEXP_EN[n] is disabled and any writes to it are ignored.
[15:0]	-	RAZ/ WI	16	-	Reserved.

4.2.3.15 AHBNSPPCEXP, AHB Non-secure Access PPC Expansion register

The AHBNSPPCEXP register permits software to configure each AHB peripheral using an AHB PPC in the expansion subsystem outside the SSE-123 Example Subsystem.

Each field defines the Secure or Non-secure access setting for an associated peripheral, as follows:

0 Permit Secure access only. Non-secure access is not permitted.
1 Permit Non-secure access only. Secure access is not permitted.

These settings directly control the expansion signals on the security control expansion interface.

The AHBNSPPCEXP register characteristics are:

Usage constraints
This register is read/write.

Configurations
This register exists in all configurations.

Attributes
See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-34: AHBNSPPCEXP register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	-	RAZ/ WI	16	-	Reserved.

Bits	Name	Access	Width	Reset value	Description
[15:0]	AHBNSPPCEXP	RW	16	`0x0000`	Expansion Non-secure access AHB slave peripheral protection control. Each bit 'n' drives the **AHBNSPPCEXP[n]**output signal. The `AHBPPCEXP_DIS` parameter defines whether each bit in this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If AHBPPCEXP_DIS[n] = `1'b1`, then AHBNSPPCEXP[n] is disabled, reads as zeros, and any writes to it are ignored.

4.2.3.16 APBNSPPC, APB Non-secure Access PPC register

The APBNSPPC register permits software to configure whether each APB peripheral that it controls using an APB *Peripheral Protection Control* (PPC) is Secure access only or is Non-secure access only.

Each field defines the Secure or Non-secure access setting for an associated peripheral, as follows:

0 Permit Secure access only. Non-secure access is not permitted.
1 Permit Non-secure access only. Secure access is not permitted.

The APBNSPPC register characteristics are:

Usage constraints
This register is read/write.

Configurations
This register exists in all configurations.

Attributes
See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-35: APBNSPPC register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:2]	-	RAZ/WI	30		Reserved.
[1]	NS_TIMER1	RW	1	`0x0`	Subsystem timer 1 APB access security setting.
[0]	NS_TIMER0	RW	1	`0x0`	Subsystem timer 0 APB access security setting.

4.2.3.17 APBNSPPCEXP0, APBNSPPCEXP1, APBNSPPCEXP2, and APBNSPPCEXP3, APB Non-secure Access PPC Expansion registers

The APBNSPPCEXPx registers permits software to configure each APB peripheral using an APB *Peripheral Protection Controller* (PPC) in the expansion subsystem outside the SSE-123 Example Subsystem.

Each field defines the Secure or Non-secure access setting for an associated peripheral, as follows:

0 Permit Secure access only. Non-secure access is not permitted.
1 Permit Non-secure access only. Secure access is not permitted.

These settings directly control the expansion signals on the security control expansion interface. All four registers have the same controls, where X is from 0-3.

The APBNSPPCEXP0, APBNSPPCEXP1, APBNSPPCEXP2, and APBNSPPCEXP3 register characteristics are:

Usage constraints

These registers are read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-36: APBNSPPCEXP0, APBNSPPCEXP1, APBNSPPCEXP2, and APBNSPPCEXP3 register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	-	RAZ/WI	16	-	Reserved.
[15:0]	APBNSPPCEXP<X>	RW	16	`0x0000`	Expansion Non-secure access APB slave peripheral protection control. Each bit 'n' drives the **APBNSPPCEXP<X>[n]** output signal and controls the selection of APBPPPCEXP<X>[n]. The `APBPPCEXP<X>_DIS` parameter defines whether each bit in this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If APBPPCEXP<X>_DIS[n] = `1'b1`, then APBNSPPCEXP<X>[n] is disabled, reads as zeros, and any writes to it are ignored.

4.2.3.18 AHBSPPPCEXP, AHB Secure Privilege Access PPC Expansion register

The AHBSPPPCEXP register permits software to configure Secure privilege access permissions of external AHB peripherals using an AHB *Peripheral Protection Controller* (PPC) located in the expansion area outside the SSE-123 Example Subsystem

Each field defines the Secure privileged or Secure unprivileged access setting for an associated peripheral, as follows:

0 Permit Secure privileged access only.
1 Permit Secure unprivileged and privileged access.

The AHBSPPPCEXP register characteristics are:

Usage constraints
This register is read/write.

Configurations
This register exists in all configurations.

Attributes
See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-37: AHBSPPPCEXP register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	-	RAZ/WI	16	-	Reserved.
[15:0]	AHBSPPPCEXP	RW	16	`0x0000`	Expansion Secure privilege access AHB slave peripheral protection control. Each bit 'n' drives the **AHBSPPPCEXP[n]** output signal. The `AHBPPCEXP_DIS` parameter defines whether each bit in this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If AHBPPCEXP_DIS[n] = `1'b1`, then AHBSPPPCEXP[n] is disabled, reads as zeros, and any writes to it are ignored.

4.2.3.19 APBSPPPC, APB Secure Privilege Access PPC register

The APBSPPPC register permits software to configure secure privilege access permissions for APB peripherals using an APB *Peripheral Protection Controller* (PPC).

Each field defines the Secure privileged or Secure unprivileged access setting for an associated peripheral, as follows:

0 Permit Secure privileged access only.

1 Permit Secure unprivileged and privileged access.

The APBSPPPC register characteristics are:

Usage constraints
This register is read/write.

Configurations
This register exists in all configurations.

Attributes
See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-38: APBSPPPC register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:7]	-	RAZ/WI	25	-	Reserved.
[6]	SP_WATCHDOG	RW	1	`0x0`	APB access Secure privileged setting for subsystem watchdog refresh frame.
[5:2]	-	RAZ/WI	4	-	Reserved.
[1]	SP_TIMER1	RW	1	`0x0`	APB access Secure privileged setting for subsystem timer 1.
[0]	SP_TIMER0	RW	1	`0x0`	APB access Secure privileged setting for subsystem timer 0.

4.2.3.20 APBSPPPCEXP0, APBSPPPCEXP1, APBSPPPCEXP2, and APBSPPPCEXP3, APB Secure Privilege Access PPC Expansion registers

The APBSPPPCEXPx register permits software to configure Secure privilege access permissions of external APB peripherals using an APB *Peripheral Protection Controller* (PPC) that are located in the expansion area outside the SSE-123 Example Subsystem.

Each field defines the Secure privileged or Secure unprivileged access setting for an associated peripheral, as follows:

0 Permit Secure privileged access only.
1 Permit Secure unprivileged and privileged access.

These settings directly control the expansion signals on the security control expansion interface. All four registers have the same controls, where X is 0-3.

The APBSPPPCEXP0, APBSPPPCEXP1, APBSPPPCEXP2, and APBSPPPCEXP3 registers characteristics are:

Usage constraints
These registers are read/write.

Configurations
This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-39: APBSPPPCEXP0, APBSPPPCEXP1, APBSPPPCEXP2, APBSPPPCEXP3 registers bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	-	RAZ/WI	16	-	Reserved.
[15:0]	APBSPPPCEXP<X>	RW	16	`0x0000`	Expansion secure privilege access APB slave peripheral protection control. Each bit 'n' drives the **APBPPPCEXP<X>[n]** output signal when the corresponding bit APBNSPPCEXP<X>[n] is set LOW. The `APBPPCEXP<X>_DIS` parameter defines whether each bit within this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If APBPPCEXP<X>_DIS[n] = `1'b1`, then APBSPPPCEXP<X>[n] is disabled, reads as zeros, and any writes to it are ignored.

4.2.3.21 NSMSCEXP, Non-secure Access MSC Expansion register

The NSMSCEXP register enables software to configure whether each master that is located behind each *Master Security Controller* (MSC) in the expansion subsystem is a Secure or Non-secure device.

The NSMSCEXP register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.3 Secure privilege control registers block on page 59.

The following table shows the bit assignments.

Table 4-40: NSMSCEXP register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	NS_MSCEXP	RW	16	`0x0000`	Expansion MSC Non-secure configuration. Each bit 'n' controls the Non-secure configuration of each MSC and drives the **NS_MSCEXP[n]** signals. Set to HIGH to define a master as Non-secure. Otherwise, it is Secure. The `MSCEXP_DIS[n]` parameter defines whether each bit in this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If MSCEXP_DIS[n] = `1'b1`, then NS_MSCEXP[n] is disabled, reads as zeros, and any writes to it are ignored.
[15:0]	-	RAZ/ WI	16	-	Reserved.

4.2.4 Non-secure privilege control registers block

The Non-secure privilege control block implements program-visible states that permit software to control various security gating units within the design.

Power domain: **nPD_SYS**.

Reset domain: **nHRESET**.

The following table shows the registers in the Non-secure privilege control block. For write access to these registers, only 32-bit writes are supported. Any byte and halfword writes result in the write data being ignored.

Table 4-41: Non-secure privilege control registers block

Offset	Name	Access	Reset value	Description	Security
`0x000`-`0x08C`		RAZ/ WI	`0x0000_0000`	Reserved.	NSP
`0x090`-`0x09C`		RAZ/ WI	`0x0000_0000`	Reserved for future Secure privileged access AHB slave peripheral protection control, AHBNSPPPC0-3.	NSP
`0x0A0`	AHBNSPPPCEXP	RW	`0x0000_0000`	4.2.4.1 AHBNSPPPCEXP, AHB Non-secure Privilege Access PPC Expansion register on page 81.	NSP
`0x0A4`-`0x0AC`		RAZ/ WI	`0x0000_0000`	Reserved, AHBNSPPPCEXP1-3.	NSP
`0x0B0`	APBNSPPPC	RW	`0x0000_0000`	4.2.4.2 APBNSPPPC, APB Non-secure Privilege Access PPC register on page 82.	NSP
`0x0B4`-`0x0BC`		RAZ/ WI	`0x0000_0000`	Reserved for future Secure privileged access APB slave peripheral protection control, APBNSPPPC1-3.	NSP
`0x0C0`	APBNSPPPCEXP0	RW	`0x0000_0000`	4.2.4.3 APBNSPPPCEXP0, APBNSPPPCEXP1, APBNSPPPCEXP2, and APBNSPPPCEXP3, APB Non-Secure Privilege PPC Expansion registers on page 83.	NSP

Offset	Name	Access	Reset value	Description	Security
`0x0C4`	APBNSPPPCEXP1	RW	`0x0000_0000`	4.2.4.3 APBNSPPPCEXP0, APBNSPPPCEXP1, APBNSPPPCEXP2, and APBNSPPPCEXP3, APB Non-Secure Privilege PPC Expansion registers on page 83.	NSP
`0x0C8`	APBNSPPPCEXP2	RW	`0x0000_0000`	4.2.4.3 APBNSPPPCEXP0, APBNSPPPCEXP1, APBNSPPPCEXP2, and APBNSPPPCEXP3, APB Non-Secure Privilege PPC Expansion registers on page 83.	NSP
`0x0CC`	APBNSPPPCEXP3	RW	`0x0000_0000`	4.2.4.3 APBNSPPPCEXP0, APBNSPPPCEXP1, APBNSPPPCEXP2, and APBNSPPPCEXP3, APB Non-Secure Privilege PPC Expansion registers on page 83.	NSP
`0x0D0`-`0xFCC`		RAZ/WI	`0x0000_0000`	Reserved.	NSP
`0xFD0`	PIDR4	RO	`0x0000_0004`	Peripheral ID 4.	NSP
`0xFD4`	-	RO	`0x0000_0000`	Reserved, Peripheral ID 5.	NSP
`0xFD8`	-	RO	`0x0000_0000`	Reserved, Peripheral ID 6.	NSP
`0xFDC`	-	RO	`0x0000_0000`	Reserved, Peripheral ID 7.	NSP
`0xFE0`	PIDR0	RO	`0x0000_0053`	Peripheral ID 0.	NSP
`0xFE4`	PIDR1	RO	`0x0000_00B8`	Peripheral ID 1.	NSP
`0xFE8`	PIDR2	RO	`0x0000_000B`	Peripheral ID 2.	NSP
`0xFEC`	PIDR3	RO	`0x0000_0000`	Peripheral ID 3.	NSP
`0xFF0`	CIDR0	RO	`0x0000_000D`	Component ID 0.	NSP
`0xFF4`	CIDR1	RO	`0x0000_00F0`	Component ID 1.	NSP
`0xFF8`	CIDR2	RO	`0x0000_0005`	Component ID 2.	NSP
`0xFFC`	CIDR3	RO	`0x0000_00B1`	Component ID 3.	NSP

4.2.4.1 AHBNSPPPCEXP, AHB Non-secure Privilege Access PPC Expansion register

The AHBSPPPCEXP register permits software to configure Non-secure privileged access permissions of external AHB peripherals using an AHB *Peripheral Protection Controller* (PPC) that is located in the expansion area outside the SSE-123 Example Subsystem.

Each field defines the Non-secure privileged or Non-secure unprivileged access setting for an associated peripheral, as follows:

0 Permit Non-secure privileged access only.
1 Permit Non-secure unprivileged and privileged access.

The AHBNSPPPCEXP register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.4 Non-secure privilege control registers block on page 80.

The following table shows the bit assignments.

Table 4-42: AHBNSPPPCEXP register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	-	RAZ/ WI	16	-	Reserved
[15:0]	AHBNSPPPCEXP	RW	16	`0x0000`	Expansion Non-secure privilege access AHB slave Peripheral Protection Control. Each bit 'n' drives the **AHBNSPPPCEXP[n]** output signal. The `AHBPPCEXP_DIS` parameter defines whether each bit within this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If AHBPPCEXP_DIS[n] = `1'b1` then AHBNSPPPCEXP[n] is disabled, reads as zeros, and any writes to it are ignored.

4.2.4.2 APBNSPPPC, APB Non-secure Privilege Access PPC register

The APBNSPPPC register permits software to configure Non-secure privilege access permissions of APB peripherals using an APB PPC.

Each field defines the Non-secure privileged or Non-secure unprivileged access setting for an associated peripheral, as follows:

`0` Permit Non-secure privileged access only.
`1` Permit Non-secure unprivileged and privileged access.

The APBNSPPPC register characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.4 Non-secure privilege control registers block on page 80.

The following table shows the bit assignments.

Table 4-43: APBNSPPPC register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:7]	-	RAZ/ WI	25	-	Reserved.
[6]	NSP_WATCHDOG	RW	1	`0x0`	APB access Non-secure privileged setting for subsystem watchdog refresh frame.

Bits	Name	Access	Width	Reset value	Description
[5:2]	-	RAZ/WI	4	-	Reserved.
[1]	NSP_TIMER1	RW	1	`0x0`	APB access Non-secure privileged setting for subsystem timer 1.
[0]	NSP_TIMER0	RW	1	`0x0`	APB access Non-secure privileged setting for subsystem timer 0.

4.2.4.3 APBNSPPPCEXP0, APBNSPPPCEXP1, APBNSPPPCEXP2, and APBNSPPPCEXP3, APB Non-Secure Privilege PPC Expansion registers

The APBNSPPPCEXPx register permits software to configure Non-secure privileged access permissions of external APB peripherals using an APB *Peripheral Protection Controller* (PPC) that is located in the expansion area outside the SSE-123 Example Subsystem.

Each field defines the Non-secure privileged or Non-secure unprivileged access setting for an associated peripheral, as follows:

0 Permit Non-secure privileged access only.
1 Permit Non-secure unprivileged and privileged access.

All four registers have the same controls, where X is from 0-3.

The APBNSPPPCEXP0, APBNSPPPCEXP1, APBNSPPPCEXP2, and APBNSPPPCEXP3 registers characteristics are:

Usage constraints

This register is read/write.

Configurations

This register exists in all configurations.

Attributes

See 4.2.4 Non-secure privilege control registers block on page 80.

The following table shows the bit assignments.

Table 4-44: APBNSPPPCEXPx register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:16]	-	RAZ/WI	16		Reserved.

Bits	Name	Access	Width	Reset value	Description
[15:0]	APBNSPPPCEXP<X>	RW	16	`0x0000`	Expansion Non-secure privilege access AHB slave Peripheral Protection Control. Each bit 'n' drives the **APBPPPCEXP<X> [n]** output signal when the corresponding APBNSPPCEXP<X>[n] bit is set LOW. The `APBPPCEXP_DIS<X>` parameter defines whether each bit within this register is implemented. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*. If APBPPCEXP_DIS<X>[n] = `1'b1` then APBNSPPPCEXP<X>[n] is disabled, reads as zeros, and any writes to it are ignored.

4.2.5 Subsystem timers

This section describes the subsystem timers in the SSE-123 Example Subsystem.

The subsystem timers are located in the following domains:

Power domain **PD_AON**.
Reset domain **nAONRESET**.

Each subsystem timer contains a single 4KB register frame that is aliased to Secure and Non-secure:

- The 4.2.3.16 APBNSPPC, APB Non-secure Access PPC register on page 75 controls the security access control of both timers.
- Both timers can be given privileged or non-privileged access.

  The following registers control the privilege level of both timers:

  - 4.2.3.19 APBSPPPC, APB Secure Privilege Access PPC register on page 77.
  - 4.2.4.2 APBNSPPPC, APB Non-secure Privilege Access PPC register on page 82.

For information about the subsystem timers registers, see B.2 System Timer on page 119.

4.2.6 Subsystem watchdogs

This section describes the subsystem watchdogs in the SSE-123 Example Subsystem.

The subsystem watchdogs are located in the following domains:

Power domain **PD_AON**.
Reset domain **nWARMAONRESET**.

Each subsystem watchdog contains two 4KB register frames that are permanently mapped to either the Secure or Non-secure:

The control frame of each watchdog is privileged access only.

The 4.2.3.19 APBSPPPC, APB Secure Privilege Access PPC register on page 77 and 4.2.4.2 APBNSPPPC, APB Non-secure Privilege Access PPC register on page 82 control whether the refresh frame of each watchdog is privileged or non-privileged access.

For more information about the subsystem watchdog registers, see B.3 System Watchdog on page 130.

4.2.7 Power Policy Unit (PPU)

This section describes the *Power Policy Unit* (PPU) in the SSE-123 Example Subsystem.

The SSE-123 Subsystem integrates CoreLink™ PCK-600 Power Control Kit PPU components to enable software to manage power control of the subsystem. See the *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual* for more information.

SYS PPU

The SYS PPU is located in the following domains:

Power domain **PD_AON**
Reset domain **nAONRESET**

Specific programming of the SYS PPU can cause the subsystem to deadlock. The programming includes:

- Any write to the PPU_PWPR.OP_DYN_EN register bit of the SYS PPU to change to a static power mode.
- Any write to the PPU.PWCR.PWR_DEVACTIVEEN register bit of the SYS PPU to disable **DEVACTIVE** inputs.

The SSE-123 Example Subsystem does not support use of the WARM_RST power mode. For information about Warm reset behavior, see 3.2.3 Reset control on page 28.

For information about SYS PPU registers, see the *Arm® Power Policy Unit Version 1.1 Architecture Specification*.

4.2.8 SRAM memory protection control

This section describes SRAM memory protection control in the SSE-123 Example Subsystem.

The SRAM memory protection control unit is located in the following domains:

Power domain **PD_SYS**.
Reset domain **nHRESET**.

For information about the SRAM MPC registers, see the *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual*.

4.2.9 Cortex-M23 processor Private Peripheral Bus (PPB)

This section describes Cortex®-M23 *Private Peripheral Bus* (PPB) in the SSE-123 Example Subsystem.

The Cortex®-M23 PPB is located in the following domains:

Power domain **PD_SYS**.
Reset domain **nCOREHRESET**.

For information about the Cortex®-M23 PPB registers, see the following:

- *Arm®v8-M Architecture Reference Manual, Chapter D1 Register Specification*.
- *Arm® Cortex®-M23 Processor Technical Reference Manual*.

4.3 Subsystem interrupt map

Interrupt sources are routed to the Cortex®-M23 processor *Nested Vector Interrupt Controller* (NVIC). All interrupts are also routed to the *Wakeup Interrupt Controller* (WIC) to be used as wakeup sources.

The options that you choose when configuring the SSE-123 Example Subsystem enable you to define the number of interrupts, and selectively enable them. You can also configure the WIC to define how many of the interrupts can be used as wakeup sources for the subsystem.

See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.

The following table shows the interrupt sources that are routed to the Cortex®-M23 processor NVIC.

Table 4-45: Subsystem interrupt map

Interrupt input	Interrupt source
NMI	Combined: • Secure watchdog interrupt. • Expansion NMI.
IRQ[0]	Non-secure watchdog reset request.
IRQ[1]	Non-secure watchdog interrupt.
IRQ[2]	Reserved, disabled.
IRQ[3]	System timer 0.
IRQ[4]	System timer 1.
IRQ[5]	Reserved, disabled.

Interrupt input	Interrupt source
IRQ[6]	CTI IRQ request 0.
IRQ[7]	CTI IRQ request 1.
IRQ[8]	Reserved, disabled.
IRQ[9]	Combined: • Subsystem SRAM MPC security violation. • Expansion MPC security violation 0-15.
IRQ[10]	Combined: • Subsystem peripherals PPC security violation. • Expansion APB PPC security violation 0-3. • Expansion AHB PPC security violation 0-3.
IRQ[11]	Combined expansion MSC security violation 0-15.
IRQ[12]	Combined expansion bridge buffer error 0-15.
IRQ[13]	Reserved, disabled.
IRQ[14]	Reserved, disabled.
IRQ[15]	**PD_SYS** PPU.
IRQ[16-239][1]	Expansion IRQ.

[1] The number of Expansion IRQ lines that are implemented depends on the configuration of the SSE-123 Example Subsystem.

Appendix A Signal descriptions

This appendix describes the SSE-123 Example Subsystem external signals.

A.1 Clock signals

The SSE-123 Example Subsystem uses a single set of standard clock signals.

The following table shows the clock signals.

Table A-1: Clock signals

Name	Direction	Power domain	Reset domain	Description
FCLK	Input	**PD_AON**	**nPORESET**	Free-running clock. This clock generates all clocks within the subsystem.
SCLK	Output	**PD_AON**	**nSYSRESET**	System domain clock. Gated when **PD_SYS** is powered OFF.
HCLK	Output	**PD_SYS**	**nHRESET**	Main bus clock. Dynamically clock gated.
DCLK	Output	**PD_SYS**	**nDBGRESET**	Debug clock. Dynamically clock gated.

A.2 Reset signals

The SSE-123 Example Subsystem uses a single set of standard reset signals.

The following table shows the reset signals.

Table A-2: Reset signals

Name	Direction	Power domain	Clock domain	Description
nPORESET	Input	**PD_AON**	**FCLK**	Active-LOW subsystem reset.
nSRST	Input	**PD_AON**	**Async**	Active-LOW debug system reset.
nAONRESET	Output	**PD_AON**	**FCLK**	Active-LOW always on domain reset.
nAONRESET_ASYNC	Output	**PD_AON**	**Async**	Active-LOW always on domain reset, not synchronized to **FCLK**.
nWARMAONRESET	Output	**PD_AON**	**FCLK**	Active-LOW warm always on domain reset.
nSYSPDRESET	Output	**PD_AON**	**Async**	Active-LOW system power domain reset.
nSYSRESET	Output	**PD_SYS**	**SCLK**	Active-LOW **SCLK** synchronized reset.
nWARMSYSRESET	Output	**PD_SYS**	**SCLK**	Active-LOW warm system power domain reset.
nHRESET	Output	**PD_SYS**	**HCLK**	Active-LOW **HCLK** synchronized reset.
nHRESET_ASYNC	Output	**PD_SYS**	**Async**	Active-LOW **HCLK** synchronized reset, not synchronized to **FCLK**.
nCOREHRESET	Output	**PD_SYS**	**HCLK**	Active-LOW **HCLK** synchronized reset for processor core, that **nSRST** can hold.
nDBGRESET	Output	**PD_SYS**	**DCLK**	Active-LOW **DCLK** synchronized reset.

A.3 Clock control interface signals

The SSE-123 Example Subsystem contains a clock control interface.

Power domain

PD_AON

Reset domain

nAONRESET

Clock domain

FCLK

The following table shows the clock control interface signals.

Table A-3: Clock control interface signals

Signal	Direction	Description
FCLKSTATEREQn	Output	Active-LOW request for high frequency clock to be supplied for **FCLK**.

A.4 HCLK clock Q-Channel signals

The SSE-123 Example Subsystem contains **HCLK** clock Q-Channel signals.

Power domain

PD_SYS

Reset domain

nWARMSYSRESET, **nHRESET**

Clock domain

SCLK, **HCLK**

The following table shows the **HCLK** clock Q-Channel signals.

Table A-4: HCLK clock Q-Channel signals

Signal	Direction	Description
HCLK_QREQn	Output	**HCLK** Q-Channel request from clock controller.
HCLK_QACCEPTn	Input	**HCLK** Q-Channel accept response to clock controller.
HCLK_QDENY	Input	**HCLK** Q-Channel deny response to clock controller.
HCLK_QACTIVE	Input	**HCLK** Q-Channel device activity indication to clock controller.

A.5 DCLK clock Q-Channel signals

The SSE-123 Example Subsystem contains **DCLK** clock Q-Channel signals.

Power domain
: **PD_SYS**

Reset domain
: **nSYSRESET**, **nDBGRESET**

Clock domain
: **SCLK**, **DCLK**

The following table shows the **DCLK** clock Q-Channel signals.

Table A-5: DCLK clock Q-Channel signals

Signal	Direction	Description
DCLK_QREQn	Output	**DCLK** Q-Channel request from clock controller.
DCLK_QACCEPTn	Input	**DCLK** Q-Channel accept response to clock controller.
DCLK_QDENY	Input	**DCLK** Q-Channel deny response to clock controller.
DCLK_QACTIVE	Input	**DCLK** Q-Channel device activity indication to clock controller.

A.6 Entry delay signals

The SSE-123 Example Subsystem contains entry delay signals.

Power domain
: **PD_AON**

Reset domain
: **nAONRESET**

Clock domain
: **FCLK**

The following table shows the entry delay signals.

Table A-6: Entry delay signals

Signal	Direction	Description
HCLK_ENTRYDELAY[7:0]	Input	Defines the number of delay cycles before the **HCLK** hierarchical clock gated domains enter the quiescence sequence.
DCLK_ENTRYDELAY[7:0]	Input	Defines the number of delay cycles before the **DCLK** hierarchical clock gated domains enter the quiescence sequence.

A.7 Clock enable signals

The SSE-123 Example Subsystem contains clock enable signals.

Power domain
    PD_AON

Reset domain
    nAONRESET

Clock domain
    FCLK

The following table shows the clock enable signals.

Table A-7: Clock enable signals

Signal	Direction	Description
SYSPPUDEVCLKEN	Output	Device clock enable from SYS PPU.

A.8 Reset control interface signals

The SSE-123 Example Subsystem contains reset control interfaces.

Reset domain
    Async

Clock domain
    Async

The following table shows the reset control interface signals.

Table A-8: Reset control interface signals

Signal	Direction	Power domain	Description
HWRESETREQ	Input	**PD_AON**	Reset request for system power domain.
HRESETDEVREQ	Output	**PD_SYS**	Device reset request for **nHRESET**.
HRESETDEVACK	Input	**PD_SYS**	Device reset request acknowledge for **nHRESET**.

A.9 Power control interfaces

The SSE-123 Example Subsystem contains the power control interfaces that this section describes.

A.9.1 PD_SYS power Q-Channel signals

The SSE-123 Example Subsystem contains **PD_SYS** power Q-Channel signals.

Power domain
: **PD_AON**

Reset domain
: **nWARMAONRESET**, **nWARMSYSRESET**, or **nHRESET**

Clock domain
: **FCLK**, **SCLK**, or **HCLK**

The following table shows the **PD_SYS** power Q-Channel signals.

Table A-9: PD_SYS power Q-Channel signals

Signal	Direction	Description
PDSYSQREQn	Output	**PD_SYS** Q-Channel request from SYS PPU.
PDSYSQACCEPTn	Input	**PD_SYS** Q-Channel accept response to SYS PPU.
PDSYSQDENY	Input	**PD_SYS** Q-Channel deny response to SYS PPU.
PDSYSQACTIVE	Input	**PD_SYS** Q-Channel device activity indication to SYS PPU.

A.9.2 Debug power interface signals

The SSE-123 Example Subsystem contains debug power interface signals.

Power domain
: **PD_AON**

Reset domain
: **nAONRESET**

Clock domain
: **Async**

The following table shows the debug power interface signals.

Table A-10: Debug power interface signals

Signal	Direction	Description
DAPPWRUPREQ	Input	Active-HIGH powerup request from debugger to subsystem power control.
DAPPWRUPACK	Output	Active-HIGH powerup acknowledge signal that indicates to the debugger that the subsystem is powered-up.

A.9.3 PPU power mode status signal

The SSE-123 Example Subsystem contains a PPU power mode status signal.

The following table shows the PPU power mode status signal.

Table A-11: PPU power mode status signal

Signal	Direction	Description
SYSPPUHWSTAT[15:0]	Output	SYS PPU current power mode.

A.10 Expansion interfaces

The SSE-123 Example Subsystem contains the expansion interfaces that this section describes.

A.10.1 AHB5 expansion master interface 0

The SSE-123 Example Subsystem contains an AHB5 expansion master interface 0.

Power domain
PD_SYS

Reset domain
nHRESET

Clock domain
HCLK

The following table shows the AHB5 expansion master interface 0 properties.

Table A-12: AHB5 expansion master interface 0 properties

Interface properties	Value	Comment
Extended_Memory_Types	TRUE	Pass-through. Not used by the subsystem.
Secure_Transfers	TRUE	Supported.
Endian	N/A	Pass-through. The bus matrix provides no built-in endian adaptation.
Stable_Between_Clock	FALSE	Not supported for the CoreLink™ SIE-200 System IP for Embedded.
Exclusive_Transfers	TRUE	Pass-through. Not used by the subsystem.
Multi_Copy_Atomicity	TRUE	No caches or buffering that make a transfer visible to only some agents.

The following table shows the AHB5 expansion master interface 0 signals.

Table A-13: AHB5 expansion master interface 0 signals

Signal	Direction	Description
HSEL_EXPM0	Output	Slave select.
HADDR_EXPM0[31:0]		Address.

Signal	Direction	Description
HTRANS_EXPM0[1:0]		Transfer type.
HWRITE_EXPM0		Transfer direction indicator.
HSIZE_EXPM0[2:0]	Output	Transfer size.
HBURST_EXPM0[2:0]		Burst type.
HPROT_EXPM0[6:0]		Protection control.
HMASTER_EXPM0[4:0]		Master identifier.
HWDATA_EXPM0[31:0]	Output	Write data.
HMASTLOCK_EXPM0		Locked sequence indicator.
HREADY_EXPM0		Transfer completion indicator from external interconnect.
HNONSEC_EXPM0		Non-secure transfer indicator.
HEXCL_EXPM0	Output	Exclusive transfer indicator.
HAUSER_EXPM0[1:0]		Address channel User signals.
HWUSER_EXPM0[1:0]		Write channel User signals.
HRDATA_EXPM0[31:0]	Input	Read data.
HREADYOUT_EXPM0		Transfer completion indicator to external interconnect or master.
HRESP_EXPM0		Transfer response.
HEXOKAY_EXPM0		Exclusive okay.
HRUSER_EXPM0[1:0]		Read channel User signals.

A.10.2 AHB5 expansion master interface 1

The SSE-123 Example Subsystem contains an AHB5 expansion master interface 1.

Power domain

PD_SYS

Reset domain

nHRESET

Clock domain

HCLK

The following table shows the AHB5 expansion master interface 1 properties.

Table A-14: AHB5 expansion master interface 1 properties

Interface properties	Value	Comment
Extended_Memory_Types	TRUE	Pass-through. Not used by the subsystem.
Secure_Transfers	TRUE	Supported.
Endian	N/A	Pass-through. The bus matrix provides no built-in endian adaptation.
Stable_Between_Clock	FALSE	Not supported for the CoreLink™ SIE-200 System IP for Embedded.
Exclusive_Transfers	TRUE	Pass-through. Not used by the subsystem.
Multi_Copy_Atomicity	TRUE	No caches or buffering that make a transfer visible to only some agents.

The following table shows the AHB5 expansion master interface 1 signals.

Table A-15: AHB5 expansion master interface 1 signals

Signal	Direction	Description
HSEL_EXPM1	Output	Slave select.
HADDR_EXPM1[31:0]		Address.
HTRANS_EXPM1[1:0]		Transfer type.
HWRITE_EXPM1		Transfer direction indicator.
HSIZE_EXPM1[2:0]	Output	Transfer size.
HBURST_EXPM1[2:0]		Burst type.
HPROT_EXPM1[6:0]		Protection control.
HMASTER_EXPM1[4:0]		Master identifier.
HWDATA_EXPM1[31:0]	Output	Write data.
HMASTLOCK_EXPM1		Locked sequence indicator.
HREADY_EXPM1		Transfer completion indicator from external interconnect.
HNONSEC_EXPM1		Non-secure transfer indicator.
HEXCL_EXPM1	Output	Exclusive transfer indicator.
HAUSER_EXPM1[1:0]		Address channel User signals.
HWUSER_EXPM1[1:0]		Write channel User signals.
HRDATA_EXPM1[31:0]	Input	Read data.
HREADYOUT_EXPM1		Transfer completion indicator to external interconnect or master.
HRESP_EXPM1		Transfer response.
HEXOKAY_EXPM1		Exclusive okay.
HRUSER_EXPM1[1:0]		Read channel User signals.

A.10.3 AHB5 expansion slave interface

The SSE-123 Example Subsystem contains an AHB5 expansion slave interface.

Power domain
: **PD_SYS**

Reset domain
: **nHRESET**

Clock domain
: **HCLK**

The following table shows the AHB5 expansion slave interface properties.

Table A-16: AHB5 expansion slave interface properties

Interface properties	Value	Comment
Extended_Memory_Types	TRUE	Pass-through. Not used by the subsystem.
Secure_Transfers	TRUE	Supported.

Interface properties	Value	Comment
Endian	N/A	Pass-through. The bus matrix provides no built-in endian adaptation.
Stable_Between_Clock	FALSE	Not supported for the CoreLink™ SIE-200 System IP for Embedded.
Exclusive_Transfers	TRUE	Pass-through. Not used by the subsystem.
Multi_Copy_Atomicity	TRUE	No caches or buffering that make a transfer visible to only some agents.

The following table shows the AHB5 expansion slave interface signals.

Table A-17: AHB5 expansion slave interface signals

Signal	Direction	Description
HSEL_EXPS0	Input	Slave select.
HADDR_EXPS0[31:0]		Address.
HTRANS_EXPS0[1:0]		Transfer type.
HWRITE_EXPS0		Transfer direction indicator.
HSIZE_EXPS0[2:0]	Input	Transfer size.
HBURST_EXPS0[2:0]		Burst type.
HPROT_EXPS0[6:0]		Protection control.
HMASTER_EXPS0[3:0]		Master identifier.
HWDATA_EXPS0[31:0]	Input	Write data.
HMASTLOCK_EXPS0		Locked sequence indicator.
HREADY_EXPS0		Transfer completion indicator from external interconnect.
HNONSEC_EXPS0		Non-secure transfer indicator.
HEXCL_EXPS0	Input	Exclusive transfer indicator.
HAUSER_EXPS0[1:0]		Address channel User signals.
HWUSER_EXPS0[1:0]		Write channel User signals.
HRDATA_EXPS0[31:0]	Output	Read data.
HREADYOUT_EXPS0		Transfer completion indicator to external interconnect or master.
HRESP_EXPS0		Transfer response.
HEXOKAY_EXPS0		Exclusive okay.
HRUSER_EXPS0[1:0]		Read channel User signals.

A.10.4 Single cycle I/O interface signals

The SSE-123 Example Subsystem contains a single cycle I/O interface.

Power domain
  PD_SYS

Reset domain
  nHRESET

Clock domain
  HCLK

The following table shows the single cycle I/O interface signals.

Table A-18: Single cycle I/O interface signals

Signal	Direction	Description
IOTRANS	Output	I/O port transaction valid.
IOADDR[31:0]	Output	I/O port transaction address.
IOWRITE	Output	I/O port transaction write control: **LOW** Transaction is a read. **HIGH** Transaction is a write.
IOWDATA[31:0]	Output	I/O port write data, for writes.
IOSIZE[1:0]	Output	I/O port transaction size: `0b00` Byte, 8-bit. `0b01` Halfword, 16-bit. `0b10` Word, 32-bit. `0b11` Never used.
IOPRIV	Output	I/O port transaction privilege level: **LOW** Non-privileged. **HIGH** Privileged.
IOMASTER	Output	I/O port transaction source: **LOW** Transaction from software on processor. **HIGH** Transaction through debug interface.
IORDATA[31:0]	Input	I/O port read data, for reads.
IONONSEC	Output	I/O port transaction security: **LOW** Transaction is Secure. **HIGH** Transaction is Non-secure.

A.11 Interrupt signals

The SSE-123 Example Subsystem contains interrupt signals.

Power domain
  PD_AON

Reset domain
  nAONRESET

Clock domain
  FCLK

The following table shows the interrupt signals.

Table A-19: Interrupt signals

Signal	Direction	Description
EXPIRQ[NUM_EXPIRQ-1:0]	Input	Expansion interrupt inputs from devices that are connected to the subsystem.
EXPNMI	Input	Expansion Non-maskable interrupt from a device that is connected to the subsystem.

A.12 Timestamp signals

The SSE-123 Example Subsystem contains timestamp signals.

The following table shows the timestamp signals.

Table A-20: Timestamp signals

Signal	Direction	Power domain	Reset domain	Clock domain	Description
TSVALUEB_SYS[63:0]	Input	**PD_AON**	**nAONRESET**	**FCLK**	System timestamp value in binary encoding.
TSVALUEB_DBG[47:0]	Input	**PD_SYS**	**nDBGRESET**	**DCLK**	Debug timestamp value in binary encoding.
TSCLKCHANGE_DBG	Input	**PD_SYS**	**nDBGRESET**	**DCLK**	Debug timestamp clock change. The clock change signal is pulsed HIGH, for one **FCLK** cycle when either the **FCLK** frequency or the timestamp counter clock frequency changes. The **TSCLKCHANGE** signal informs the trace tools that any previously inferred frequency relationships might have changed, and must therefore be recalculated for future reference. **Note:** Because this process takes some time, it does not matter if the pulse timing varies by a few cycles from the actual change of clock frequency.

A.13 Event signals

The SSE-123 Example Subsystem contains event signals.

Reset domain
: **nAONRESET**

Clock domain
: **FCLK**

The following table shows the event signals.

Table A-21: Event signals

Signal	Direction	Power domain	Description
RXEV	Input	**PD_AON**	A HIGH on this input causes the Arm® v8-M architecture-defined Event Register to be set in the Cortex®-M23 processor and a WFE instruction to complete. It also wakes the processor if it is sleeping as the result of encountering a WFE instruction when the Event Register is clear.
TXEV	Output	**PD_SYS**	A single **FCLK** cycle pulse is generated on this output every time an SEV instruction is executed on the Cortex®-M23 processor.

A.14 Debug interfaces

The SSE-123 Example Subsystem contains the debug interfaces that this section describes.

A.14.1 Debug authentication signals

The SSE-123 Example Subsystem contains debug authentication signals.

Power domain
: **PD_AON**

Reset domain
: **nAONRESET**

Clock domain
: **FCLK**

The following table shows the debug authentication signals.

Table A-22: Debug authentication signals

Signal	Direction	Description
DBGENIN	Input	Invasive debug enable input.
NIDENIN	Input	Non-invasive debug enable input.
SPIDENIN	Input	Secure invasive debug enable input.
SPNIDENIN	Input	Secure non-invasive debug enable input.
DBGEN	Output	Invasive debug enable output.
NIDEN	Output	Non-invasive debug enable output.
SPIDEN	Output	Secure invasive debug enable output.
SPNIDEN	Output	Secure non-invasive debug enable output.
DBGEN_SEL_DIS	Input	**DBGEN** selector disable. When set HIGH, disables the **DBGEN** selector logic and forces **DBGEN** to use **DBGENIN**.
NIDEN_SEL_DIS	Input	**NIDEN** selector disable. When set HIGH, disables the **NIDEN** selector logic and forces **NIDEN** to use **NIDENIN**.

Signal	Direction	Description
SPIDEN_SEL_DIS	Input	**SPIDEN** selector disable. When set HIGH, disables the **SPIDEN** selector logic and forces **SPIDEN** to use **SPIDENIN**.
SPNIDEN_SEL_DIS	Input	**SPNIDEN** selector disable. When set HIGH, disables the **SPNIDEN** selector logic and forces **SPNIDEN** to use **SPNIDENIN**.

A.14.2 Debug slave interface signals

The SSE-123 Example Subsystem contains debug slave interface signals.

Power domain
  PD_SYS

Reset domain
  nDBGRESET

Clock domain
  DCLK

The following table shows the debug slave interface signals.

Table A-23: Debug slave interface signals

Signal	Direction	Description
HRDATA_SDBG[31:0]	Output	Read data.
HREADY_SDBG	Output	Transfer completion indicator.
HRESP_SDBG	Output	Transfer response.
HADDR_SDBG[31:0]	Input	Address.
HTRANS_SDBG[1:0]	Input	Transfer type.
HWRITE_SDBG	Input	Transfer direction indicator.
HWDATA_SDBG[31:0]	Input	Write data.
HSIZE_SDBG[1:0]	Input	Size of the transfer.
HPROT_SDBG[6:0]	Input	Protection control.
HNONSEC_SDBG	Input	Non-secure transfer indicator.

A.14.3 ATB ETM slave interface signals

The SSE-123 Example Subsystem contains ATB ETM slave interface signals.

Power domain
  PD_SYS

Reset domain
  nDBGRESET

Clock domain

DCLK

The following table shows the ATB ETM slave interface signals.

Table A-24: ATB ETM slave interface signals

Signal	Direction	Description
ATREADY_ETM	Input	ETM is ready to accept data.
ATDATA_ETM[7:0]	Output	Trace data.
ATVALID_ETM	Output	A transfer is valid during this cycle. If LOW, all other AT signals must be ignored during this cycle.
AFREADY_ETM	Output	ETM FIFO has been flushed.
ATID_ETM[6:0]	Output	Identifies the source of the trace.
AFVALID_ETM	Input	ETM FIFO flush request.

A.14.4 ETM control signals

The SSE-123 Example Subsystem contains *Embedded Trace Macrocell* (ETM) control signals.

The following table shows the ETM control signals.

Table A-25: ETM control signals

Signal	Direction	Description
ETMFIFOFULLEN	Input	Indicates support for FIFOFULL functionality by reading an ETM register.
ETMEN	Output	Set by the trace software tools to ensure that the trace output is enabled from the ETM.
ETMTRIGOUT	Output	Indicates a trigger packet in the trace stream.

A.14.5 Cross Trigger Interface signals

The SSE-123 Example Subsystem contains *Cross Trigger Interface* (CTI) signals.

Power domain

PD_SYS

Reset domain

nDBGRESET

Clock domain

DCLK

The following table shows the CTI signals.

Table A-26: CTI signals

Signal	Direction	Description
CTICHIN[3:0]	Input	CTI channel in.

Signal	Direction	Description
CTICHOUT[3:0]	Output	CTI channel out.

A.14.6 APB debug expansion master interface signals

The SSE-123 Example Subsystem contains APB debug expansion master interface signals.

Power domain
: **PD_SYS**

Reset domain
: **nDBGRESET**

Clock domain
: **DCLK**

The following table shows the APB debug expansion master interface signals.

Table A-27: APB debug expansion master interface signals

Signal	Direction	Description
PADDR_MDBG[31:0]	Output	Address.
PSEL_MDBG	Output	Peripheral select.
PENABLE_MDBG	Output	Enable for transfer.
PWRITE_MDBG	Output	Write transaction indicator.
PWDATA_MDBG[31:0]	Output	Write data.
PREADY_MDBG	Input	Transfer ready.
PRDATA_MDBG[31:0]	Output	Read data.
PSLVERR_MDBG	Input	Error response.

A.15 Security control expansion signals

The SSE-123 Example Subsystem contains security control expansion signals.

Power domain
: **PD_SYS**

Reset domain
: **nHRESET**

Clock domain
: **HCLK**

The following table shows the security control expansion signals.

Table A-28: Security control expansion signals

Signal	Direction	Description
ACCWAITN	Output	This request signal controls any external gating unit that might be required to block accesses to the system using the AHB slave expansion interfaces: `0` No gating. `1` Access gated. The 4.2.3.2 BUSWAIT, Bus Access Wait register on page 62 controls this signal.
ACCWAITN_STAT	Input	This status signal indicates the current state of any external gating unit that might be used to block access to the system using the AHB slave expansion interfaces: `0` No gating. `1` Access gated. The 4.2.3.2 BUSWAIT, Bus Access Wait register on page 62 can read this signal.
SECRESPCFG	Output	This signal configures how to respond to an access when a security violation occurs. `0` Read-Zero Write Ignore. `1` Bus error. The 4.2.3.3 SECRESPCFG, Security Violation Response Configuration register on page 63 controls this signal.
SMPCEXP_STAT[15:0]	Input	Interrupt status inputs from all expansion memory protection controller. These inputs are visible to the programmer using the 4.2.3.5 SECMPCINTSTATUS, Secure MPC Interrupt Status register on page 65 and raise an interrupt using the MPC combined interrupt. The `MPCEXP_DIS` top-level parameter enables each individual bit of the interface to be disabled. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.
APBPPCEXP_STAT[3:0]	Input	APB PPC interrupt status input. Each bit is to be connected to a single APB PPC. These bits are associated with the S_APBPPCEXP_STATUS field in the 4.2.3.6 SECPPCINTSTAT, Secure PPC Interrupt Status register on page 66, 4.2.3.7 SECPPCINTCLR, Secure PPC Interrupt Clear register on page 66, and 4.2.3.8 SECPPCINTEN, Secure PPC Interrupt Enable register on page 67.
APBPPCEXP_CLR[3:0]	Output	APB PPC interrupt clear output. Each bit is to be connected to a single APB PPC. These bits are associated with the S_APBPPCEXP_CLR field in the 4.2.3.6 SECPPCINTSTAT, Secure PPC Interrupt Status register on page 66, 4.2.3.7 SECPPCINTCLR, Secure PPC Interrupt Clear register on page 66, and 4.2.3.8 SECPPCINTEN, Secure PPC Interrupt Enable register on page 67.
APBNSPPCEXP0[15:0] **APBNSPPCEXP1[15:0]** **APBNSPPCEXP2[15:0]** **APBNSPPCEXP3[15:0]**	Output	APB PPC Non-secure gating control. Each bit is to be connected to a PPC to control Non-secure access for an APB interface. The 4.2.3.17 APBNSPPCEXP0, APBNSPPCEXP1, APBNSPPCEXP2, and APBNSPPCEXP3, APB Non-secure Access PPC Expansion registers on page 75 drive these signals. The `APBPPCEXP_DIS<X>` top-level parameter enables individual bits to be disabled. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.
APBPPPCEXP0[15:0] **APBPPPCEXP1[15:0]** **APBPPPCEXP2[15:0]** **APBPPPCEXP3[15:0]**	Output	APB PPC Privilege Gating Control. Each bit is to be connected to a PPC to control privilege access for an APB interface. The values of **APBNSPPPCEXP<X>** or **APBSPPPCEXP<X>** drive these signals. Selection of the source register for these signals depends on the bits set in the 4.2.3.17 APBNSPPCEXP0, APBNSPPCEXP1, APBNSPPCEXP2, and APBNSPPCEXP3, APB Non-secure Access PPC Expansion registers on page 75. The `APBPPCEXP_DIS<X>` top-level parameter enables individual bits to be disabled. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.

Signal	Direction	Description
SAHBPPCEXP_STAT[3:0]	Input	AHB PPC interrupt status input. Each bit is to be connected to a single AHB PPC. These bits are associated with the S_AHBPPCEXP_STATUS field in the 4.2.3.6 SECPPCINTSTAT, Secure PPC Interrupt Status register on page 66, 4.2.3.7 SECPPCINTCLR, Secure PPC Interrupt Clear register on page 66, and 4.2.3.8 SECPPCINTEN, Secure PPC Interrupt Enable register on page 67.
SAHBPPCEXP_CLR[3:0]	Output	AHB PPC interrupt clear output. Each bit is to be connected to a single AHB PPC. These bits are associated with the S_AHBPPCEXP_CLR field in the 4.2.3.6 SECPPCINTSTAT, Secure PPC Interrupt Status register on page 66, 4.2.3.7 SECPPCINTCLR, Secure PPC Interrupt Clear register on page 66, and 4.2.3.8 SECPPCINTEN, Secure PPC Interrupt Enable register on page 67.
AHBNSPPCEXP[15:0]	Output	AHB PPC Non-secure gating control. Each bit is to be connected to a PPC to control Non-secure access for an AHB interface. The 4.2.3.15 AHBNSPPCEXP, AHB Non-secure Access PPC Expansion register on page 74 drives these signals. The `AHBPPCEXP_DIS` top-level parameter enables individual bits to be disabled. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.
AHBPPPCEXP[15:0]	Output	APB PPC privilege gating control. Each bit is to be connected to a PPC to control privilege access for an APB interface. The 4.2.3.18 AHBSPPPCEXP, AHB Secure Privilege Access PPC Expansion register on page 76 drives these signals. The `AHBPPCEXP_DIS` top-level parameter enables individual bits to be disabled. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.
SMSCEXP_STAT[15:0]	Input	MSC interrupt status input. Each bit is to be connected to a single MSC. These bits are associated with the S_MSCEXP_STATUS field in the 4.2.3.9 SECMSCINTSTAT, Secure MSC Interrupt Status register on page 68, 4.2.3.10 SECMSCINTCLR, Secure MSC Interrupt Clear register on page 69, and 4.2.3.11 SECMSCINTEN, Secure MSC Interrupt Enable register on page 70. The `MSCEXP_DIS` top-level parameter enables each individual bit of the interface to be disabled. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.
SMSCEXP_CLR[15:0]	Output	MSC interrupt clear output. Each bit is to be connected to a single MSC. These bits are associated with the S_MSCEXP_CLR field in the 4.2.3.9 SECMSCINTSTAT, Secure MSC Interrupt Status register on page 68, 4.2.3.10 SECMSCINTCLR, Secure MSC Interrupt Clear register on page 69, and 4.2.3.11 SECMSCINTEN, Secure MSC Interrupt Enable register on page 70. The `MSCEXP_DIS` top-level parameter enables each individual bit of the interface to be disabled. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.
NSMSCEXP[15:0]	Output	MSC Non-secure configuration. Each bit is to be connected to a single MSC. The NS_MSCEXP field in the 4.2.3.21 NSMSCEXP, Non-secure Access MSC Expansion register on page 79 drives these signals. The `MSCEXP_DIS` top-level parameter enables each individual bit of the interface to be disabled. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.

Signal	Direction	Description
BRGEXP_STAT[15:0]	Input	Bridge error interrupt status input. Each bit is to be connected to a single bridge. These bits are associated with the BRGEXP_STATUS field in the 4.2.3.12 BRGINTSTAT, Bridge Interrupt Status register on page 71, 4.2.3.13 BRGINTCLR, Bridge Interrupt Clear register on page 72, and 4.2.3.14 BRGINTEN, Bridge Interrupt Enable register on page 73. The `BRGEXP_DIS` top-level parameter enables each individual bit of the interface to be disabled. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.
BRGEXP_CLR[15:0]	Output	Bridge error interrupt clear output. Each bit is to be connected to a single bridge. These bits are associated with the BRGEXP_CLEAR field in the 4.2.3.12 BRGINTSTAT, Bridge Interrupt Status register on page 71, 4.2.3.13 BRGINTCLR, Bridge Interrupt Clear register on page 72, and 4.2.3.14 BRGINTEN, Bridge Interrupt Enable register on page 73. The `BRGEXP_DIS` top-level parameter enables each individual bit of the interface to be disabled. See the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.

A.16 Static configuration signals

The SSE-123 Example Subsystem contains static configuration signals.

The following table shows the static configuration signals.

Table A-29: Static configuration signals

Signal	Direction	Description
INITVTOR_RST[23:0]	Input	Default Secure vector table offset at reset. Sets the reset value for the 4.2.2.10 INITVTOR, Initial Secure Vector Table Offset register on page 56.
CPUWAIT_RST	Input	Default CPUWAIT value at reset. Sets the reset value of the 4.2.2.11 CPUWAIT, CPU Wait register on page 57.

A.17 System control signals

The SSE-123 Example Subsystem contains system control signals.

Power domain
: **PD_AON**

Reset domain
: **nPORESET**

Clock domain
: **Async**

The following table shows the system control signals.

Table A-30: System control signals

Signal	Direction	Description
CPUWAIT_CLR	Input	The HIGH level of **CPUWAIT_CLR** detected by the subsystem clears the 4.2.2.11 CPUWAIT, CPU Wait register on page 57 once, enabling the processor to boot. This input can be asynchronous to the subsystem. If it is asynchronous, it can be used with the **CPUWAIT_CLR_R** output as a 4-phase handshake to ensure capture across domains.
CPUWAIT_CLR_R	Output	The status of **CPUWAIT_CLR** after synchronization to the subsystem.

A.18 System status signals

The SSE-123 Example Subsystem contains system status signals.

The following table shows the system status signals.

Table A-31: System status signals

Signal	Direction	Description
WICSENSE[241:0]	Output	Active-HIGH set of signals. These signals indicate which input lines can cause the *Wakeup Interrupt Controller* (WIC) to wakeup the subsystem.
HALTED	Output	Indicates that the processor is halted.

A.19 DFT interface signals

The SSE-123 Example Subsystem contains DFT interface signals.

The following table shows the DFT interface signals.

Table A-32: DFT interface signals

Signal	Direction	Description
DFTCGEN	Input	DFT clock gate enable.
DFTISODISABLE	Input	DFT isolation disable. Use **DFTISODISABLE** to disable isolation of all the internal power domains.
DFTPWRUP	Input	DFT powerup. Use **DFTPWRUP** to force all the internal power domains to turn on.
DFTRSTDISABLE	Input	DFT reset disable.
DFTRAMHOLD	Input	DFT RAM hold.

Appendix B System time components

Three system time components are delivered with the SSE-123 Example Subsystem product and integrated as an example in the SSE-123 Subsystem and SSE-123 Integration.

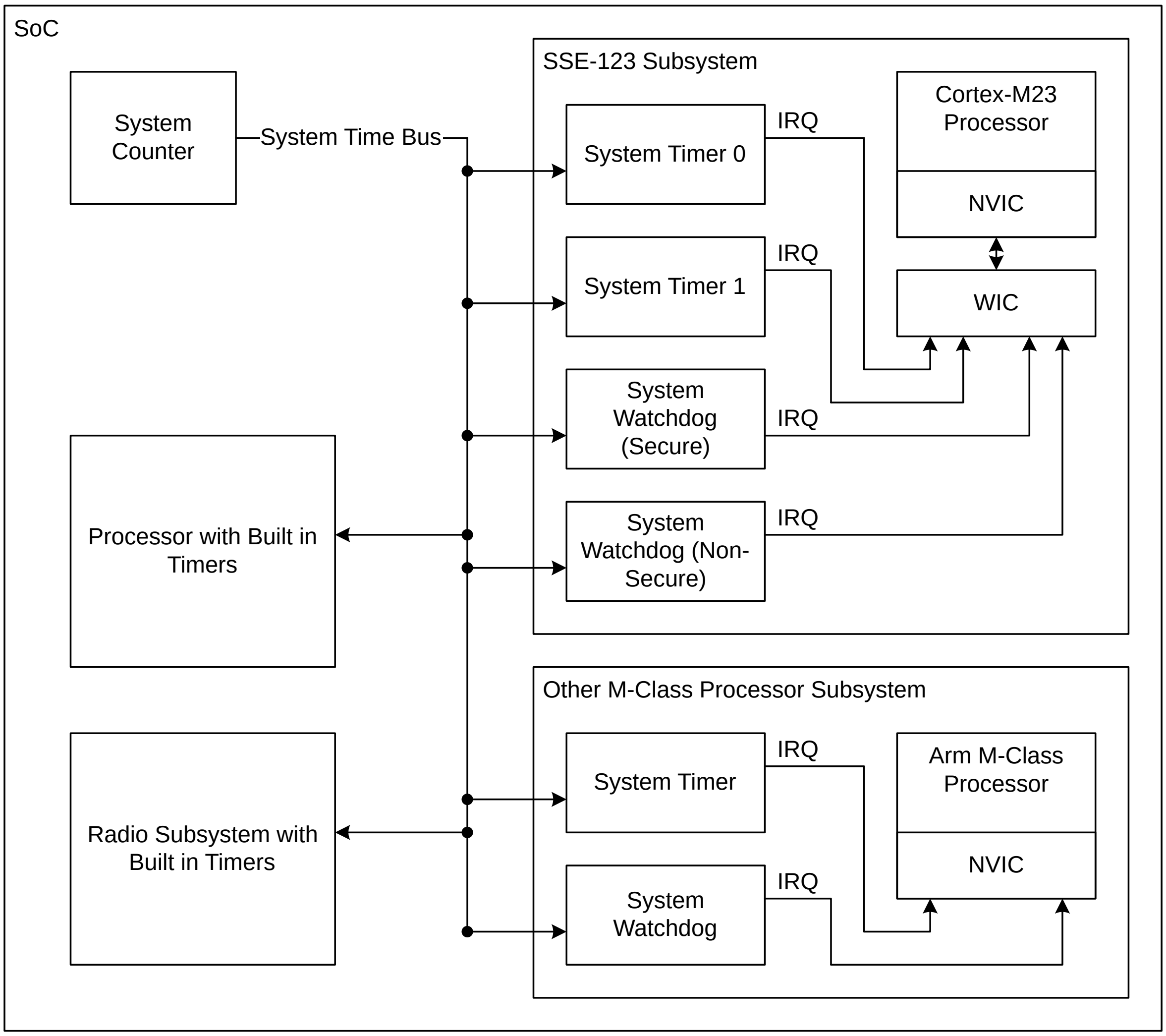
The system time components are as follows:

- The System Counter generates a timestamp value that can be shared across the *System on Chip* (SoC).
- The System Timer can raise an interrupt when a period has elapsed.
- System Watchdog provides a mechanism to detect errant system behavior causing reset of the system if a period elapses without intervention.

The components are software-programmable using APB interfaces. This section defines the programmers model of the components.

The following figure shows an example SoC that uses the system time components.

Figure B-1: Example SoC that uses the system time components



B.1 System Counter

The SSE-123 Example Subsystem contains a system counter.

B.1.1 System Counter overview

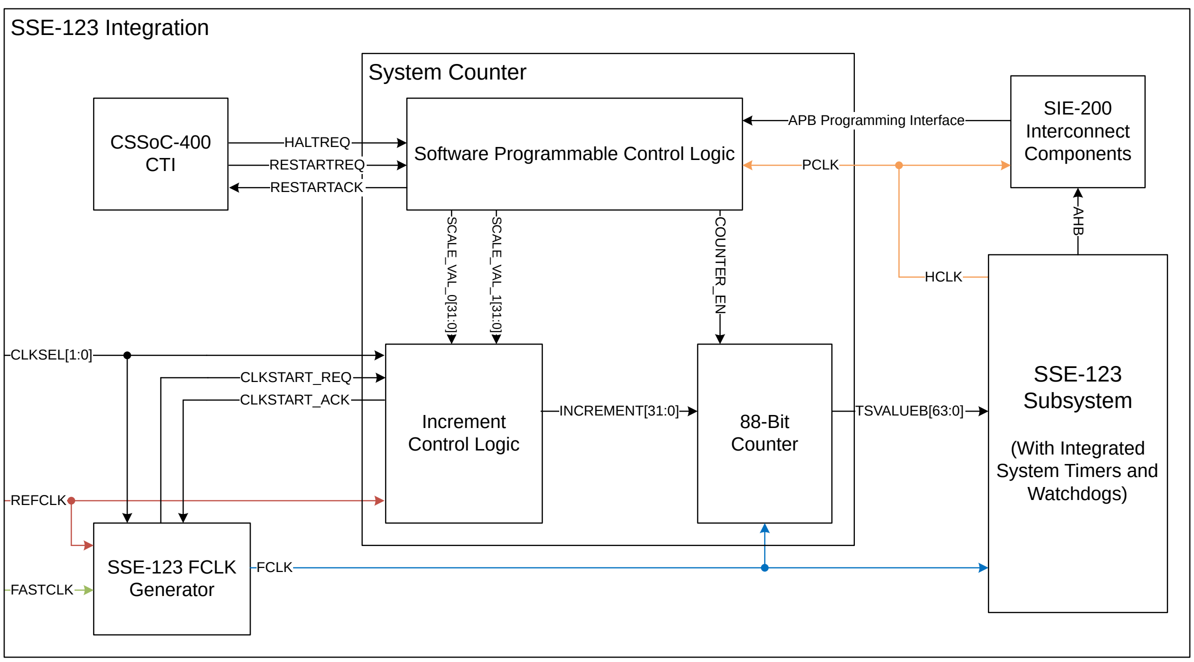
This section provides an overview of the System Counter.

The System Counter has the following features:

- Binary encoded, unsigned, 64-bit up-counter, starts counting from 0, with the optional ability to start counting from a different preload value.
- Generates a 64-bit time value that compatible components across the SoC can share.
- Internal 24-bit fractional value to allow fine count resolution control.
- Ability to scale counter clock frequency, therefore allowing operation at lower frequency during low-power mode.
- Hardware can handle dynamic clock switching between different frequencies. Software is required only for initialization.
- Support of software enabled halt-on-debug from external CoreSight™ *Cross Trigger Interface* (CTI).

The following figure shows a block diagram of the System Counter that is integrated within the SSE-123 Integration.

Figure B-2: System Counter integrated within the SSE-123 Integration



For more information, see the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual*.

B.1.2 Counter operation

This section describes pseudo-sequences for some of the commonly used Counter programming flows.

Counter initialization

To initialize the counter, perform the following steps:

1. Ensure that **CLKSEL** is `0x01` and that **REFCLK** is running.
2. Disable the counter by setting CNTCR.EN = 0.
3. Configure the SoC clock generation to generate the frequencies that are required for the two clock sources.
4. Write to CNTSCR0 to set the scaling value for **REFCLK** clock source.
5. Write to CNTSCR1 to set the scaling value for **FASTCLK** clock source.
6. Enable the counter by setting CNTCR.EN = 1.
7. **CLKSEL** can now be changed at any time to switch the **FCLK** source between **REFCLK** and **FASTCLK**.

Changing scaling registers

To change the scaling registers, perform the following steps:

1. Ensure that **REFCLK** is running.
2. Disable the counter by setting CNTCR.EN = 0.
3. Write new values to CNTSCR0 and CNTSCR1.
4. Enable the counter by setting CNTCR.EN = 1.

B.1.3 Programmers model

This section describes programmers model for the System Counter.

Registers in the System Counter provide the following functions:

- Enabling and disabling the counter.
- Setting the counter value.
- Changing the operating mode, to change the frequency and increment value.
- Enabling Halt-on-debug, that a debugger can then use to suspend counting.
- Providing status of the Counter value in addition to the operating mode.

These registers are grouped into two 4KB frames:

- A control frame, CNTControlBase.
- A status frame, CNTReadBase.

The base addresses of these frames are **IMPLEMENTATION DEFINED**. Similarly, the security level of each frame is also **IMPLEMENTATION DEFINED**, however, in a system that supports both, Secure and Non-secure memory maps, CNTControlBase is only accessible by Secure memory accesses.

See the SSE-123 Integration Programmers Model in the *Arm® SSE-123 Example Subsystem Configuration and Integration Manual* for the base addresses of these registers in the SSE-123 Integration.

B.1.3.1 CNTControlBase registers summary

This section provides a summary of the CNTControlBase Frame Registers (System Counter).

The following table shows a summary of the CNTControlBase Frame Registers (System Counter)

Table B-1: CNTControlBase Frame Registers (System Counter)

Offset	Name	Access	Reset value	Description
`0x000`	CNTCR	RW	`0x0000_0000`	Counter Control Register. See B.1.3.3.1 CNTCR, Counter Control register on page 113.
`0x004`	CNTSR	RO	`0x0000_000X`	Counter Status Register. See B.1.3.3.2 CNTSR, Counter Status register on page 113.
`0x008`	CNTCV[31:0]	RW	`0xXXXX_XXXX`	Counter Count Value register. See B.1.3.3.3 CNTCV, Counter Count Value register on page 114.
`0x00C`	CNTCV[63:32]	RW	`0xXXXX_XXXX`	
`0x010`	CNTSCR[2]	RW	`0x0100_0000`	Counter Scale Register. See B.1.3.3.4 CNTSCR, Counter Scale register on page 114.
`0x014-0x018`	-	RAZ/WI	-	Reserved.
`0x01C`	CNTID	RO	`0x000X_000X`	Counter ID register. See B.1.3.3.5 CNTID, Counter ID register on page 115.
`0x020-0x0BC`	-	RAZ/WI	-	Reserved.
`0x0C0-0x0CC`	-	RAZ/WI	-	Reserved.
`0x0D0`	CNTSCR0[2]	RW	`0x0100_0000`	Counter Scale Register 0. See B.1.3.3.6 CNTSCR0, CNTSCR1, Counter Scaling registers on page 116.
`0x0D4`	CNTSCR1	RW	`0x0100_0000`	Counter Scale Register 1. See B.1.3.3.6 CNTSCR0, CNTSCR1, Counter Scaling registers on page 116.
`0x0D8-0xFCC`	-	RAZ/WI	-	Reserved.
`0xFD0`	CNTPIDR4	RO	`0x0000_0004`	B.1.3.3.7 CNTPIDR4, Peripheral Identification Register 4 on page 116.
`0xFD4-0xFDC`	-	RAZ/WI	-	Reserved.
`0xFE0`	CNTPIDR0	RO	`0x0000_00BA`	B.1.3.3.8 CNTPIDR0, Peripheral Identification Register 0 on page 117.
`0xFE4`	CNTPIDR1	RO	`0x0000_00B0`	B.1.3.3.9 CNTPIDR1, Peripheral Identification Register 1 on page 117.
`0xFE8`	CNTPIDR2	RO	`0x0000_000B`	B.1.3.3.10 CNTPIDR2, Peripheral Identification Register 2 on page 117.
`0xFEC`	CNTPIDR3	RO	`0x0000_0000`	B.1.3.3.11 CNTPIDR3, Peripheral Identification Register 3 on page 117.

Offset	Name	Access	Reset value	Description
`0xFF0`	CNTCIDR0	RO	`0x0000_000D`	B.1.3.3.12 CNTCIDR0, Component Identification Register 0 on page 118.
`0xFF4`	CNTCIDR1	RO	`0x0000_00F0`	B.1.3.3.13 CNTCIDR1, Component Identification Register 1 on page 118.
`0xFF8`	CNTCIDR2	RO	`0x0000_0005`	B.1.3.3.14 CNTCIDR2, Component Identification Register 2 on page 118.
`0xFFC`	CNTCIDR3	RO	`0x0000_00B1`	B.1.3.3.15 CNTCIDR3, Component Identification Register 3 on page 119.

B.1.3.2 CNTReadBase registers summary

This section provides a summary of the CNTReadBase registers.

The following table shows a summary of the CNTReadBase Frame Registers (System Counter).

Table B-2: CNTReadBase Frame Registers (System Counter)

Offset	Name	Access	Reset value	Description
`0x000`	CNTCV[31:0]	RO	`0xXXXX_XXXX`	Counter Count Value register. See B.1.3.3.3 CNTCV, Counter Count Value register on page 114.
`0x004`	CNTCV[63:32]	RO	`0xXXXX_XXXX`	
`0x008-0xFCC`	-	RAZ/WI	-	Reserved.
`0xFD0`	CNTPIDR4	RO	`0x0000_0004`	B.1.3.3.7 CNTPIDR4, Peripheral Identification Register 4 on page 116.
`0xFD4-0xFDC`	-	RAZ/WI	-	Reserved.
`0xFE0`	CNTPIDR0	RO	`0x0000_00BB`	B.1.3.3.8 CNTPIDR0, Peripheral Identification Register 0 on page 117.
`0xFE4`	CNTPIDR1	RO	`0x0000_00B0`	B.1.3.3.9 CNTPIDR1, Peripheral Identification Register 1 on page 117.
`0xFE8`	CNTPIDR2	RO	`0x0000_000B`	B.1.3.3.10 CNTPIDR2, Peripheral Identification Register 2 on page 117.
`0xFEC`	CNTPIDR3	RO	`0x0000_0000`	B.1.3.3.11 CNTPIDR3, Peripheral Identification Register 3 on page 117.
`0xFF0`	CNTCIDR0	RO	`0x0000_000D`	B.1.3.3.12 CNTCIDR0, Component Identification Register 0 on page 118.
`0xFF4`	CNTCIDR1	RO	`0x0000_00F0`	B.1.3.3.13 CNTCIDR1, Component Identification Register 1 on page 118.
`0xFF8`	CNTCIDR2	RO	`0x0000_0005`	B.1.3.3.14 CNTCIDR2, Component Identification Register 2 on page 118.
`0xFFC`	CNTCIDR3	RO	`0x0000_00B1`	B.1.3.3.15 CNTCIDR3, Component Identification Register 3 on page 119.

[2] CNTSCR is aliased with CNTSCR0, meaning that either addresses of CNTSCR and CNTSCR0 physically access a single register.

B.1.3.3 Register descriptions

This section describes each System Counter register.

All registers are 32-bit and must be accessed using 32-bit reads and writes.

B.1.3.3.1 CNTCR, Counter Control register

The CNTCR register enables the counter, controls the counter frequency setting, and controls counter behavior during debug.

The following table shows the bit assignments.

Table B-3: CNTCR register bit assignments

Bits	Name	Access	Reset value	Function
[31:6]	-	RAZ/WI	`0x0000`	Reserved.
[5]	INTRCLR	RW	`0x0`	Interrupt clear bit, only writes of 0 are permitted, and writes of 1 are ignored. If APB logic is powered off when the interrupt output is asserted, this bit is cleared automatically. Therefore, it is the responsibility of software to ensure that there is no pending interrupt before powering off the APB logic.
[4]	PSLVERRDIS	RW	`0x0`	**PSLVERR** output disable: **0** **PSLVERR** permanently driven to '0'. **1** **PSLVERR** output that the System Counter generates dynamically.
[3]	INTRMASK	RW	`0x0`	Interrupt mask: **0** Interrupt output disabled. **1** Interrupt output enabled.
[2]	SCEN	RW	`0x0`	Scale enable: **0** Scaling is not enabled. The Counter value is incremented by `0x01.000000` for each counter tick. **1** Scaling is enabled. The counter is incremented by the ScaleVal for each counter tick. The ScaleVal value that the System Counter uses is from CNTSCR, CNTSCR[0], or CNTSCR[1]. See B.1.3.3.6 CNTSCR0, CNTSCR1, Counter Scaling registers on page 116.
[1]	HDBG	RW	`0x0`	Halt On Debug: **0** **HALTREQ** signal into the Counter has no effect. **1** **HALTREQ** signal into the Counter halts the Count.
[0]	EN	RW	`0x0`	Enable Counter: **0** Disabled: Count is not incrementing. **1** Enabled: Count is incrementing.

B.1.3.3.2 CNTSR, Counter Status register

The CNTSR register provides Counter frequency status information.

The following table shows the bit assignments.

Table B-4: CNTSR register bit assignments

Bits	Name	Access	Reset value	Function
[31:2]	-	RAZ/WI.	`0x000000`	Reserved.
[1]	DBGH	RO	UNK	Indicates whether the counter is halted because the Halt-on-Debug signal is asserted: **0** Counter is not halted. **1** Counter is halted.
[0]	-	RAZ/WI.	`0x0`	Reserved.

B.1.3.3.3 CNTCV, Counter Count Value register

The CNTCV register indicates the current count value.

The following table shows the bit assignments.

Table B-5: CNTCV register bit assignments

Bits	Name	Access	Function
[63:0]	CountValue	RO from CNTReadBase. RW from CNTControlBase.	Indicates the count value.

B.1.3.3.4 CNTSCR, Counter Scale register

The CNTSCR registers store the Counter Scaling value.

The following table shows the bit assignments.

Table B-6: CNTSCR register bit assignments

Bits	Name	Access	Reset value	Function
[31:0]	ScaleVal	RW	`0x0100_0000`[3]	When counter scaling is enabled, ScaleVal is the amount added to the Counter Count Value for every period of the counter as determined by 1/Frequency from the current operating frequency of the system counter, the *counter tick*. ScaleVal is expressed as an unsigned fixed-point number with an 8-bit integer value and a 24-bit fractional value. The CNTSCR register can only be changed when the counter is disabled, that is, CNTCR.EN=0. If the value of CTNSCR changes when CNTCR.EN==1, then the Counter Count Value becomes **UNKNOWN** and remains **UNKNOWN** on future ticks of the clock.

[3] If CNTSC=0, CNTSCR is permanently driven to `0x0100_0000`.

B.1.3.3.5 CNTID, Counter ID register

The CNTID register indicates additional information about Counter Scaling implementation.

The following table shows the bit assignments.

Table B-7: CNTID register bit assignments

Bits	Name	Access	Reset value	Function
[31:20]	-	RAZ/WI	`0x000`	Reserved.
[19]	CNTSCR_OVR	RO	Defined by parameter with a default value of `0x0`	Override counter enable condition for writing to CNTSCR* registers: **`0`** CNTSCR* can be written only when CNTCR.EN=0. **`1`** CNTSCR* can be written when CNTCR.EN=0 or 1.
[18:17]	CNTSELCLK	RO	`0x01`	Indicates the clock source that the Counter is using. Based on the settings for Counter scaling, the Counter increment value is chosen either from one of the CNTSCR registers or is fixed to 1.0 when scaling is disabled: **`00`** Invalid status, Counter not incrementing. **`01`** CLK0 (**REFCLK**). **`10`** CLK1 (**FASTCLK**). **`11`** Invalid status, counter not incrementing. These bits are only valid when hardware clock switching is implemented (**HWCLKSW**=1). If **HWCLKSW**=0, these bits read a constant value of `0x01` regardless of the value of **TSCLKSEL** input and Counter increment value is used from CNTSCR0 or fixed to 1.0 depending on the status of Counter scaling feature.
[16]	CNTCS	RO	Defined by parameter with a default value of `0x1`	Indicates whether Clock Switching is implemented: **`0`** HW-based Counter Clock Switching is not implemented. **`1`** HW-based Counter Clock Switching is implemented.
[15:4]	-	RAZ/WI	0x000	Reserved
[3:0]	CNTSC	RO	`0x1`	Indicates whether Counter Scaling is implemented: **`0000`** Counter Scaling is not implemented. **`0001`** Counter Scaling is implemented. All other values are Reserved.

Note

CNTCS and CNTSELSC are new fields that are defined in this implementation.

B.1.3.3.6 CNTSCR0, CNTSCR1, Counter Scaling registers

The CNTSCR0 and CNTSCR1 registers have the same field definitions as the CNTSCR register. These two extra registers are used to preprogram the scaling values so that when hardware-based clock switching is implemented there is no need to program the scaling increment value each time when clock is switched.

When read by software, the value read is the value that software has written. In certain cases, for example when CNTCR.SCEN has disabled scaling, the actual increment value that the Counter uses is 1.0. However, the value that is read from these registers does not reflect this. Therefore, the actual increment value that the Counter uses depends on the programming of these registers, the value of two HW configuration parameters, `CNTSC` and `HWCLKSW`, and the value of CNTCR.SCEN.

This implementation enables software to keep the scaling values in these registers unaffected when increment value changes due to Counter disabling.

These registers can only be written when the Counter is disabled (CNTCR.EN=0).

The following table shows the bit assignments.

Table B-8: CNTSCR* register bit assignments

Bits	Name	Access	Reset value	Function
[31:0]	ScaleVal	RW	`0x0100_0000`[4]	When counter scaling is enabled, ScaleVal is the amount added to the Counter Count Value for every period of the counter as determined by 1/Frequency from the current operating frequency of the system counter, the *counter tick*. ScaleVal is expressed as an unsigned fixed-point number with an 8-bit integer value and a 24-bit fractional value. The CNTSCR register can only be changed when the counter is disabled, which CNTCR.EN=0. If the value of CTNSCR changes when CNTCR.EN==1, then the Counter Count Value becomes **UNKNOWN** and remains **UNKNOWN** on future ticks of the clock.

B.1.3.3.7 CNTPIDR4, Peripheral Identification Register 4

The CNTPIDR4 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-9: CNTPIDR4 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	SIZE	`0x0`	The 4KB count.
[3:0]	DES_2	`0x4`	Indicates the JEP106 continuation code.

[4] If CNTSC=0, CNTSCR* are permanently driven to `0x0100_0000`.

If HWCLKSW= 0, CNTSCR1 is permanently driven to `0x0000_0000`.

B.1.3.3.8 CNTPIDR0, Peripheral Identification Register 0

The CNTPIDR0 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-10: CNTPIDR0 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PART_0	`0xBA` - CNTControlBase `0xBB` - CNTReadBase	Indicates the part number [7:0].

B.1.3.3.9 CNTPIDR1, Peripheral Identification Register 1

The CNTPIDR1 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-11: CNTPIDR1 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	DES_0	`0xB`	The JEP106 identity code [3:0].
[3:0]	PART_1	`0x0`	The part number [11:8].

B.1.3.3.10 CNTPIDR2, Peripheral Identification Register 2

The CNTPIDR2 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-12: CNTPIDR2 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	REVISION	`0x0`	The revision.
[3]	JEDEC	`0x1`	Set to the required value of 1.
[2:0]	DES_1	`0x3`	Indicates the JEP106 continuation code.

B.1.3.3.11 CNTPIDR3, Peripheral Identification Register 3

The CNTPIDR3 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-13: CNTPIDR3 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	REVAND	`0x0`	The manufacturer revision number.
[3:0]	CMOD	`0x0`	Incremented on authorized customer modification.

B.1.3.3.12 CNTCIDR0, Component Identification Register 0

The CNTCIDR0 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-14: CNTCIDR0 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PRMBL_0	`0x0D`	The CNTCIDR[7:0].

B.1.3.3.13 CNTCIDR1, Component Identification Register 1

The CNTCIDR1 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-15: CNTCIDR1 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	CLASS	`0xF`	The CNTCIDR[15:12].
[3:0]	PRMBL_1	`0x0`	The CNTCIDR[11:8].

B.1.3.3.14 CNTCIDR2, Component Identification Register 2

The CNTCIDR2 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-16: CNTCIDR2 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PRMBL_2	`0x05`	The CNTCIDR[23:16].

B.1.3.3.15 CNTCIDR3, Component Identification Register 3

The CNTCIDR3 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-17: CNTCIDR3 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PRMBL_3	`0xB1`	The CNTCIDR[31:24].

B.2 System Timer

The SSE-123 Example Subsystem contains a system counter.

B.2.1 System Timer overview

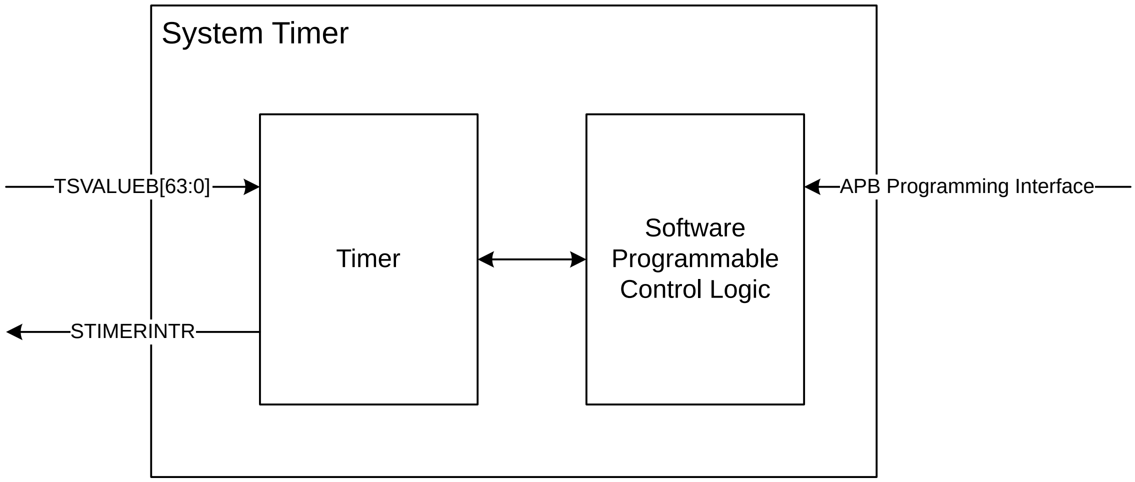
This section provides an overview of the System Timer.

The System Timer has the following features:

- Count up and count down timer functionality with *auto-increment* feature.
- Generation of a level triggered interrupt after a preconfigured period of time has passed.
- Time-based on shared system time count that the System Counter generates.

The following figure shows a block diagram of the System Timer.

Figure B-3: System Timer block diagram



B.2.2 System Timer operation

The primary function of the System Timer is to generate an interrupt output that is based on the Timer configuration and interrupt mask setting.

The timers can be programmed to count:

- Up to a threshold, by programming a CompareValue (`CNTP_CVAL`).
- Down from a programmed value, by programming a TimerValue (`CNTP_TVAL`).

The basic function of the System Timer is:

```
TimerCondMet = Counter[63:0] - CompareValue[63:0] >= 0
```

An alternative 32-bit view, called TimerValue, can be used to set the CompareValue as follows:

```
CompareValue = (Counter[63:0] + SignExtend(TimerValue)[63:0]
```

And reading the TimerValue gives the count down value:

```
TimerValue = (CompareValue[31:0] - Counter [31:0]
```

The *auto-increment* feature allows generation of Timer interrupt at regular intervals without the need for reprogramming the Timer after each interrupt and re-enabling the timer logic.

The Automatic Increment timer view operates as a 64-bit up-counter. An AutoIncrValue Reload register is programmed with a 32-bit offset.

If the EN bit of the AutoIncrValue Control register is set, the offset value is zero-extended, summed with the count value, and loaded into the AutoIncrValue register, performed automatically by hardware.

The operation of *auto-increment* is as follows:

```
AutoIncrValue = (ZeroExtend(Reload[31:0])[63:0] + Counter[63:0])
```

Where:

`AutoIncrValue`	The value of the AutoIncrValue register.
`Reload`	The value of the AutoIncrValue Reload register.
`Counter`	The value of the Count register.

The timer condition is met when the count value reaches the value that is loaded into its AutoIncrValue register:

```
TimerCondMet = (((Counter[63:0] - AutoIncrValue[63:0]) >= 0)
```

Where:

`TimerCondMet TRUE`	If the timer condition is met.
`TimerCondMet FALSE`	Otherwise.

When the timer condition is met:

- An interrupt is generated, if the interrupt is not masked in the timer control register, and remains asserted until software clears it by writing to the CLR bit of the AutoIncrValue Control register, CNTP_AIVAL_CTL).
- The CLR bit in the AutoIncrValue Control register is set to 1 and it remains 1 until software clears it by writing '0'.
- The AutoIncrValue register is reloaded with the new value.

When the *auto-increment* feature is enabled, the operation of normal timer, using the compare value or timer value registers, is disabled. This is to ensure that when the interrupt is generated, the cause of interrupt is unambiguous to the user.

The auto-increment timer starts counting only when both the Timer is enabled (CNTP_CTL.ENABLE=1) and *auto-increment* mode is enabled by setting CNTP_AIVAL_CTL.EN=1. When both are enabled, the Timer starts counting down until the `AIVAL_RELOAD` period is reached.

B.2.3 Programmers model

The registers of the System Timer are grouped into a single 4KB block that is called the CNTBase frame. The base address of the CNTBase frame is not defined here and is **IMPLEMENTATION DEFINED**.

See 4. Programmers model on page 42 for the base addresses of these registers in the SSE-123 Subsystem.

B.2.3.1 CNTBase Registers Summary

This section provides a summary of the CNTBase Registers.

The following table shows a summary of the CNTBase Registers.

Table B-18: CNTBase Frame Registers summary

Offset	Name	Access	Reset value	Description
0x000	CNTPCT[31:0]	RO	0xXXXX_XXXX	Physical Count Register. See B.2.3.2.1 CNTPCT, Counter-timer Physical Count register on page 123.
0x004	CNTPCT[63:32]	RO	0xXXXX_XXXX	
0x008-0x00C	-	-	-	Reserved.
0x010	CNTFRQ[5]	RW	0x0000_0000	Counter Frequency register. See B.2.3.2.2 CNTFRQ, Counter-timer Frequency register on page 123.
0x014-0x01C	-	-	-	Reserved.
0x020	CNTP_CVAL[31:0]	RW	0x0000_0000	Timer CompareValue register. See B.2.3.2.3 CNTP_CVAL, Counter-timer Physical Timer CompareValue register on page 124.
0x024	CNTP_CVAL[63:32]	RW	0x0000_0000	
0x028	CNTP_TVAL	RW	0xXXXX_XXXX	TimerValue register. See B.2.3.2.4 CNTP_TVAL, Counter-timer Physical Timer TimerValue register on page 124.
0x02C	CNTP_CTL	RW	0x0000_000X	Timer Control register. See B.2.3.2.5 CNTP_CTL, Counter-timer Physical Timer Control register on page 125.
0x030-0x03C	-	-	-	Reserved.
0x040	CNTP_AIVAL[31:0]	RO	0xXXXX_XXXX	AutoIncrValue register. See B.2.3.2.6 CNTP_AIVAL, AutoIncrValue register on page 126.
0x044	CNTP_AIVAL[63:32]	RO	0xXXXX_XXXX	
0x048	CNTP_AIVAL_RELOAD	RW	0xXXXX_XXXX	AutoIncrValue Reload register. See B.2.3.2.7 CNTP_AIVAL_RELOAD, AutoIncrValue Reload register on page 126.
0x04C	CNTP_AIVAL_CTL	RW	0xXXXX_XXXX	AutoIncrValue Control register. See B.2.3.2.8 CNTP_AIVAL_CTL, AutoIncrValue Control register on page 126.
0x050	CNTP_CFG	RO	0x0000_0001	Timer Configuration register. See B.2.3.2.9 CNTP_CFG, Configuration register on page 127.
0x054-0xFCC	-	-	-	Reserved.

[5] The ARMv8-A defines the CNTFRQ register to enable software to discover the frequency of Timer clock. This register is included for software-compatibility but is not used in this implementation where the Counter can have hardware-switchable clocks.

Offset	Name	Access	Reset value	Description
`0xFD0`	CNTP_PID4	RO	`0x0000_0004`	B.2.3.2.10 CNTP_PID4, Peripheral Identification Register 4 on page 127.
`0xFD4-0xFDC`	-	RO	-	Reserved.
`0xFE0`	CNTP_PID0	RO	`0x0000_00B7`	B.2.3.2.11 CNTP_PID0, Peripheral Identification Register 0 on page 127.
`0xFE4`	CNTP_PID1	RO	`0x0000_00B0`	B.2.3.2.12 CNTP_PID1, Peripheral Identification Register 1 on page 128.
`0xFE8`	CNTP_PID2	RO	`0x0000_000B`	B.2.3.2.13 CNTP_PID2, Peripheral Identification Register 2 on page 128.
`0xFEC`	CNTP_PID3	RO	`0x0000_0000`	B.2.3.2.14 CNTP_PID3, Peripheral Identification Register 3 on page 128.
`0xFF0`	CNTP_CID0	RO	`0x0000_000D`	B.2.3.2.15 CNTP_CID0, Component Identification Register 0 on page 129.
`0xFF4`	CNTP_CID1	RO	`0x0000_00F0`	B.2.3.2.16 CNTP_CID1, Component Identification Register 1 on page 129.
`0xFF8`	CNTP_CID2	RO	`0x0000_0005`	B.2.3.2.17 CNTP_CID2, Component Identification Register 2 on page 129.
`0xFFC`	CNTP_CID3	RO	`0x0000_00B1`	B.2.3.2.18 CNTP_CID3, Component Identification Register 3 on page 130.

B.2.3.2 Register descriptions

This section describes each of the System Timer registers.

B.2.3.2.1 CNTPCT, Counter-timer Physical Count register

The CNTPCT register holds the 64-bit physical count value.

The following table shows the bit assignments.

Table B-19: CNTPCT register bit assignments

Bits	Name	Access	Reset value	Function
[63:0]	CountValue	RO	`0xXXXX_XXXX_XXXX_XXXX`	Physical count value.

B.2.3.2.2 CNTFRQ, Counter-timer Frequency register

The CNTFRQ register is provided so that software can discover the frequency of the system counter. The instance of this register in the CNTCTLBase frame must be programmed with this value as part of system initialization. Hardware does not interpret the value of the register.

The following table shows the bit assignments.

Table B-20: CNTFRQ register bit assignments

Bits	Name	Access	Reset value	Function
[31:0]	ClockFrequency	RW	`0x0000_0000`	Clock frequency.

B.2.3.2.3 CNTP_CVAL, Counter-timer Physical Timer CompareValue register

The CNTP_CVAL register holds the 64-bit compare value for the timer.

The following table shows the bit assignments.

Table B-21: CNTP_CVAL register bit assignments

Bits	Name	Access	Reset value	Function
[63:0]	CompareValue	RW	`0x0000_0000_0000_0000`	Holds the 64-bit compare value for the timer. When CNTP_CTL.ENABLE is 1, the timer condition is met when (CNTPCT - CompareValue) is greater than zero. This means that CompareValue acts like a 64-bit upcounter timer. When the timer condition is met: • CNTP_CTL.ISTATUS is set to 1. • An interrupt is generated if CNTP_CTL.IMASK is 0. When CNTP_CTL.ENABLE is 0, the timer condition is not met, but CNTPCT continues to count.

B.2.3.2.4 CNTP_TVAL, Counter-timer Physical Timer TimerValue register

The CNTP_TVAL register holds the timer value for the timer.

The following table shows the bit assignments.

Table B-22: CNTP_TVAL register bit assignments

Bits	Name	Access	Reset value	Function
[31:0]	TimerValue	RW	`0xXXXX_XXXX`	The TimerValue view of the physical timer. On a read of this register: • If CNTP_CTL.ENABLE is 0, the value that is returned is **UNKNOWN**. • If CNTP_CTL.ENABLE is 1, the value that is returned is (CNTP_CVAL - CNTPCT). On a write of this register, CNTP_CVAL is set to (CNTPCT + TimerValue), where TimerValue is treated as a signed 32-bit integer. When CNTP_CTL.ENABLE is 1, the timer condition is met when (CNTPCT - CNTP_CVAL) is greater than zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met: • CNTP_CTL.ISTATUS is set to 1. • If CNTP_CTL.IMASK is 0, an interrupt is generated. When CNTP_CTL.ENABLE is 0, the timer condition is not met, but CNTPCT continues to count, so the TimerValue view appears to continue to count down.

B.2.3.2.5 CNTP_CTL, Counter-timer Physical Timer Control register

The CNTP_CTL register is a control register for the timer.

The following table shows the bit assignments.

Table B-23: CNTP_CTL register bit assignments

Bits	Name	Access	Reset value	Function
[31:3]	-	RAZ/WI	-	Reserved.
[2]	ISTATUS	RO	`0xX`	The status of the timer. This bit indicates whether the timer condition is met: **0** Timer condition is not met. **1** Timer condition is met. When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0, then the timer interrupt is asserted. When the value of the ENABLE bit is 0, the ISTATUS field is **UNKNOWN**.
[1]	IMASK	RW	`0xX`	Timer interrupt mask bit. Permitted values are: **0** The IMASK bit masks the Timer interrupt. **1** The IMASK bit masks the Timer interrupt. For more information, see the description of the ISTATUS bit.

Bits	Name	Access	Reset value	Function
[0]	ENABLE	RW	`0x0`	Enables the timer. Permitted values are: **0** Timer is disabled. **1** Timer is enabled. Setting this bit to 0 disables the timer output signal, but the timer value accessible from CNTP_TVAL continues to count down. Disabling the output signal might be a power-saving option.

B.2.3.2.6 CNTP_AIVAL, AutoIncrValue register

The CNTP_AIVAL register holds the 64-bit Automatic Increment value for the timer.

The following table shows the bit assignments.

Table B-24: CNTP_AIVAL register bit assignments

Bits	Name	Access	Width	Reset value	Description
[63:0]	AutoIncrValue timer value	RO	64	`0x0000_0000_0000_0000`	Timer AutoIncrValue.

B.2.3.2.7 CNTP_AIVAL_RELOAD, AutoIncrValue Reload register

Holds the programmable offset value for the Automatic Increment timer view.

The following table shows the bit assignments.

Table B-25: CNTP_AIVAL_RELOAD register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:0]	AutoIncrValue Reload value	RW	32	`0x0000_0000_0000_0000`	Timer AutoIncrValue Reload.

B.2.3.2.8 CNTP_AIVAL_CTL, AutoIncrValue Control register

Control register for the Automatic Increment timer view.

The following table shows the bit assignments.

Table B-26: CNTP_AIVAL_CTL register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:2]	--	-	30	-	Reserved.

Bits	Name	Access	Width	Reset value	Description
[1]	CLR	RW	1	`0xX`	Timer interrupt clear bit. This bit is only used to clear the interrupt that is generated because of auto-increment mode. For clearing the interrupt generated by normal mode, the Timer should be disabled. Permitted values are: **0** Timer interrupt is clear. **1** Timer interrupt is not clear. The CLR bit can only be written to as zero. Writes as one are ignored.
[0]	EN	RW	1	`0xX`	Enables the AutoIncrValue register. Permitted values are: **0** AutoIncrValue disabled. **1** AutoIncrValue enabled.

B.2.3.2.9 CNTP_CFG, Configuration register

Provides timer configuration information.

The following table shows the bit assignments.

Table B-27: CNTP_CFG register bit assignments

Bits	Name	Access	Width	Reset value	Description
[31:4]	-	-	28	-	Reserved.
[3:0]	AIVAL	RO	4	`0x1`	Indicates whether Automatic Increment is implemented. Permitted values are: **`0000`** Automatic Increment is not implemented. **`0001`** Automatic Increment is implemented. All other values are Reserved.

B.2.3.2.10 CNTP_PID4, Peripheral Identification Register 4

The CNTP_PID4 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-28: CNTP_PID4 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	SIZE	`0x0`	The 4KB count.
[3:0]	DES_2	`0x4`	Indicates the JEP106 continuation code.

B.2.3.2.11 CNTP_PID0, Peripheral Identification Register 0

The CNTPIDR0 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-29: CNTP_PID0 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PART_0	`0xB7`	Indicates the part number [7:0].

B.2.3.2.12 CNTP_PID1, Peripheral Identification Register 1

The CNTP_PID1 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-30: CNTP_PID1 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	DES_0	`0xB`	The JEP106 identity code [3:0].
[3:0]	PART_1	`0x0`	The part number [11:8].

B.2.3.2.13 CNTP_PID2, Peripheral Identification Register 2

The CNTP_PID2 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-31: CNTP_PID2 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	REVISION	`0x0`	The revision.
[3]	JEDEC	`0x1`	Set to the required value of 1.
[2:0]	DES_1	`0x3`	Indicates the JEP106 continuation code.

B.2.3.2.14 CNTP_PID3, Peripheral Identification Register 3

The CNTP_PID3 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-32: CNTP_PID3 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	REVAND	`0x0`	The manufacturer revision number.
[3:0]	CMOD	`0x0`	Incremented on authorized customer modification.

B.2.3.2.15 CNTP_CID0, Component Identification Register 0

The CNTP_CID0 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-33: CNTP_CID0 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PRMBL_0	`0x0D`	The Component ID0[7:0].

B.2.3.2.16 CNTP_CID1, Component Identification Register 1

The CNTP_CID1 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-34: CNTP_CID1 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	CLASS	`0xF`	The Component ID3[15:12].
[3:0]	PRMBL_1	`0x0`	The Component ID3[11:8].

B.2.3.2.17 CNTP_CID2, Component Identification Register 2

The CNTP_CID2 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-35: CNTP_CID2 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PRMBL_2	`0x05`	The Component ID2[23:16].

B.2.3.2.18 CNTP_CID3, Component Identification Register 3

The CNTP_CID3 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-36: CNTP_CID3 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PRMBL_3	`0xB1`	The Component ID1[31:24].

B.3 System Watchdog

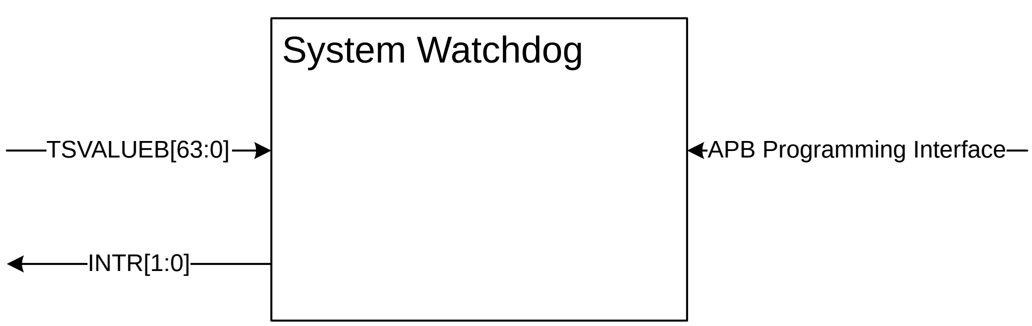
The System Watchdog aids the detection of errant system behavior. If the System Watchdog is not refreshed periodically, it raises a signal, that is typically wired to an interrupt. If this watchdog remains unrefreshed, it raises a second signal that can be used to interrupt higher-privileged software or cause a reset.

B.3.1 System Watchdog overview

This section provides an overview of the System Watchdog.

The following figure shows a block diagram of the System Watchdog.

Figure B-4: Block diagram of the System Watchdog



B.3.2 Watchdog operation

The basic function of the Generic Watchdog is to count for a fixed period of time, during which it expects to be refreshed by the system, indicating normal operation.

If a refresh occurs within the watch period, the period is refreshed to the start.

If the refresh does not occur, then the watch period expires, and a signal **INTR[0]** is raised, and a second watch period is begun.

The initial signal is typically wired to an interrupt and alerts the system. The system can attempt to take corrective action that includes refreshing the watchdog within the second watch period. If the refresh is successful, the system returns to the previous normal operation.

If it fails, then the second watch period expires, and a second signal **INTR[1]** is generated. The signal is fed to higher privileged software as an interrupt or reset for it to take executive action.

The Watchdog uses the input time `TSVALUEB` generated by the System Counter as the timebase against which the decision to trigger an interrupt is made.

B.3.3 Programmers model

This section describes the programmers model of the System Watchdog.

The Watchdog includes the following two 4KB register frames:

- Control Frame.
- Refresh Frame.

Configurable parameters control the base addresses of Control and Refresh frames. These parameters provide the flexibility of arranging these two frames within an 8KB memory map that the following table shows. See 4. Programmers model on page 42 for the base addresses of these registers in SSE-123 Subsystem.

The following table shows the Base addresses of Control and Refresh frames.

Table B-37: Base addresses of Control and Refresh frames

Description	Start address	End address	Size
Control Frame	`0x0_0000`	`0x0_0FFF`	4KB
Refresh Frame	`0x0_1000`	`0x0_1FFF`	4KB

B.3.3.1 Control Frame registers summary

This section provides a summary of the Control Frame registers.

The following table shows the Control Frame registers summary.

Table B-38: Control Frame registers summary

Offset	Name	Access	Reset value	Description
0x000	WCS	RW	0x0000_0000	B.3.3.3.1 WCS, Watchdog Control and Status register on page 133
0x004	-	RES0	-	Reserved.
0x008	WOR	RW	0x0000_0000	B.3.3.3.2 WOR, Watchdog Offset register on page 133.
0x00c	-	RES0	-	Reserved.
0x010	WCV[31:0]	RW	0x0000_0000	B.3.3.3.3 WCV, Watchdog Compare Value register on page 133
0x014	WCV[63:32]	RW	0x0000_0000	-
0x018-0xFC8	-	RES0	-	Reserved.
0xFCC	W_IIDR	RO	0x0000_143B	B.3.3.3.5 W_IIDR, Watchdog Interface Identification register on page 134
0xFD0		RO	0x0000_0004	B.3.3.3.6 PID4, Peripheral ID Register 4 on page 134
0xFE0		RO	0x0000_00B1	B.3.3.3.7 PID0, Peripheral ID Register 0 on page 135
0xFE4		RO	0x0000_00B0	B.3.3.3.8 PID1, Peripheral ID Register 1 on page 135
0xFE8		RO	0x0000_002B	B.3.3.3.9 PID2, Peripheral ID Register 2 on page 135
0xFEC		RO	0x0000_0000	B.3.3.3.10 PID3, Peripheral ID Register 3 on page 136
0xFF0		RO	0x0000_000D	B.3.3.3.11 CID0, Component ID Register 0 on page 136
0xFF4		RO	0x0000_00F0	B.3.3.3.12 CID1, Component ID Register 1 on page 136
0xFF8		RO	0x0000_0005	B.3.3.3.13 CID2, Component ID Register 2 on page 137
0xFFC		RO	0x0000_00B1	B.3.3.3.14 CID3, Component ID Register 3 on page 137

B.3.3.2 Refresh Frame registers Summary

This section provides a summary of the Refresh Frame registers.

The following table shows a summary of the Refresh Frame registers.

Table B-39: Refresh Frame registers summary

Offset	Name	Access	Reset value	Description
0x000	WRR	RW	0x0000_0000	B.3.3.3.4 WRR, Watchdog Refresh register on page 134.
0x004-0xFC8	-	-	-	Reserved.
0xFCC	W_IIDR	RO	0x0000_143B	B.3.3.3.5 W_IIDR, Watchdog Interface Identification register on page 134.
0xFD0	PID4	RO	0x0000_0004	B.3.3.3.6 PID4, Peripheral ID Register 4 on page 134
0xFE0	PID0	RO	0x0000_00B0	B.3.3.3.7 PID0, Peripheral ID Register 0 on page 135
0xFE4	PID1	RO	0x0000_00B0	B.3.3.3.8 PID1, Peripheral ID Register 1 on page 135
0xFE8	PID2	RO	0x0000_002B	B.3.3.3.9 PID2, Peripheral ID Register 2 on page 135
0xFEC	PID3	RO	0x0000_0000	B.3.3.3.10 PID3, Peripheral ID Register 3 on page 136

Offset	Name	Access	Reset value	Description
`0xFF0`	CID0	RO	`0x0000_000D`	B.3.3.3.11 CID0, Component ID Register 0 on page 136
`0xFF4`	CID1	RO	`0x0000_00F0`	B.3.3.3.12 CID1, Component ID Register 1 on page 136
`0xFF8`	CID2	RO	`0x0000_0005`	B.3.3.3.13 CID2, Component ID Register 2 on page 137
`0xFFC`	CID3	RO	`0x0000_00B1`	B.3.3.3.14 CID3, Component ID Register 3 on page 137

B.3.3.3 Register descriptions

This section describes each System Watchdog register.

All registers are 32-bit and must be accessed using 32-bit reads and writes.

B.3.3.3.1 WCS, Watchdog Control and Status register

The WCS register is a control and status register for the watchdog. Any write to this register causes an explicit watchdog refresh.

The following table shows the bit assignments.

Table B-40: WCS register bit assignments

Bits	Name	Access	Reset value	Function
[31:3]	-	RAZ/WI	-	Reserved.
[2:1]	Watchdog Signal Status	RO	`0x0`	Indicates the current state of the watchdog signals: **Bit 0** WS0 Interrupt INTR[0]. **Bit 1** WS1 Interrupt INTR[1].
[0]	Watchdog Enable	RW	`0x0`	Watchdog enable: **0** A write of 0 disables the Watchdog. **1** A write of 1 to this bit enables the Watchdog. A read of these bits indicates the current state of the Watchdog enable.

B.3.3.3.2 WOR, Watchdog Offset register

The WOR register is a countdown timer value for the watchdog. Any write to this register causes an explicit watchdog refresh.

The following table shows the bit assignments.

Table B-41: WOR register bit assignments

Bits	Name	Access	Reset value	Function
[31:0]	WOR	RW	`0x0000_0000`	Holds the 32-bit countdown timer value.

B.3.3.3.3 WCV, Watchdog Compare Value register

The WCV register holds the compare value of the watchdog.

The following table shows the bit assignments.

Table B-42: WCV register bit assignments

Bits	Name	Access	Reset value	Function
[63:0]	WCV	RW	`0x0000_0000`	Holds the current 64-bit compare value.

B.3.3.3.4 WRR, Watchdog Refresh register

The WRR register is a refresh register for the Watchdog. Any write to this register causes an explicit watchdog refresh.

The following table shows the bit assignments.

Table B-43: WRR register bit assignments

Bits	Name	Access	Function
[31:0]	WRR	RW	A write to this location causes the Watchdog to refresh and start a new watch period. A read has no effect and returns 0.

B.3.3.3.5 W_IIDR, Watchdog Interface Identification register

The W_IIDR register is an identification register for the watchdog.

The following table shows the bit assignments.

Table B-44: W_IIDR register bit assignments

Bits	Name	Access	Reset value	Function
[31:24]	ID	RO	`0x00`	Product identifier. `0x00` = System Watchdog.
[23:20]	-	RO	`0x0`	Reserved.
[19:16]	ARCH	RO	`0x0`	Architecture version, v0.
[15:12]	REV	RO	`0x1`	Revision number for the component.
[11:0]	JEPCODE	RO	`0x43B`	Arm® JEP106 code.

B.3.3.3.6 PID4, Peripheral ID Register 4

The PID4 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-45: PID4 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	SIZE	`0x0`	The 4KB count.
[3:0]	DES_2	`0x4`	Indicates the JEP106 continuation code.

B.3.3.3.7 PID0, Peripheral ID Register 0

The PIDR0 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-46: PIDR0 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PART_0	`0xB0` - Refresh Frame `0xB1` - Control Frame	Indicates the part number [7:0].

B.3.3.3.8 PID1, Peripheral ID Register 1

The PID1 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-47: PID1 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	DES_0	`0xB`	The JEP106 identity code [3:0].
[3:0]	PART_1	`0x0`	The part number [11:8].

B.3.3.3.9 PID2, Peripheral ID Register 2

The PID2 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-48: PID2 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.

Bits	Name	Default	Function
[7:4]	REVISION	`0x0`	The revision.
[3]	JEDEC	`0x2`	Set to the required value of 2.
[2:0]	DES_1	`0xB`	Indicates the JEP106 continuation code.

B.3.3.3.10 PID3, Peripheral ID Register 3

The PID3 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-49: PID3 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	REVAND	`0x0`	The manufacturer revison number.
[3:0]	CMOD	`0x0`	Incremented on authorized customer modification.

B.3.3.3.11 CID0, Component ID Register 0

The CID0 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-50: CID0 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PRMBL_0	`0x0D`	The Component ID0[7:0].

B.3.3.3.12 CID1, Component ID Register 1

The CID1 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-51: CID1 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:4]	CLASS	`0xF`	The CNTCIDR[15:12].
[3:0]	PRMBL_1	`0x0`	The Component ID1[11:8].

B.3.3.3.13 CID2, Component ID Register 2

The CID2 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-52: CID2 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PRMBL_2	`0x05`	The Component ID2[23:16].

B.3.3.3.14 CID3, Component ID Register 3

The CID3 is a read-only register that provides standard information. Only bits [7:0] are used and the remaining bits are Reserved.

The following table shows the bit assignments.

Table B-53: CID3 register bit assignments

Bits	Name	Default	Function
[31:8]	-	`0x0`	Reserved.
[7:0]	PRMBL_3	`0xB1`	The Component ID3[31:24].

Appendix C Revisions

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

Table C-1: Issue 0000-00

Change	Location
First release	-

Table C-2: Differences between issue 0000-00 and issue 0000-01

Change	Location
Added details for the PID and CID registers.	B.1.3 Programmers model on page 110

Table C-3: Differences between issue 0000-01 and issue 0000-02

Change	Location
Corrected confidentiality of Power Control System Architecture Specification, and Clock Controller Architecture Specification, from Non-Confidential to Confidential.	1.4 Additional reading on page 11
Corrected spellings of APBSPPPCEXP0, APBSPPPCEXP1, APBSPPPCEXP2, and APBSPPPPCEXP3 register names in Secure privilege control registers block summary table.	Table 4-19: Secure privilege control registers block on page 60
Corrected register access type descriptions.	4.2.2.3 SECDBGCLR, Secure Debug Clear register on page 51 4.2.2.4 SCSECCTRL, System Control Security Control register on page 52 4.2.2.6 RESET_SYNDROME, Reset Syndrome register on page 53 4.2.2.7 RESET_MASK, Reset Mask register on page 54 4.2.2.8 SWRESET, Software Reset register on page 55 4.2.3.1 SPCSECCTRL, Secure Privilege Controller Secure Configuration Control register on page 62
Corrected register access type description. Corrected list describing actions taken for the assigned bit[1] and bit[0] values, to include those assigned values.	B.2.3.2.8 CNTP_AIVAL_CTL, AutoIncrValue Control register on page 126
Corrected memory regions size of region ID 22 in subsystem memory map table.	Table 4-1: Subsystem memory region descriptions on page 44