Arm® Corstone™ Reference Systems

Revision: r0p0

Architecture Specification Ma2

Non-Confidential

Issue 01

107610_0000_01_en

Arm® Corstone™ Reference Systems
Architecture Specification Ma2

Release information

Document history

Issue	Date	Confidentiality	Change
0000-01	31 August 2023	Non-Confidential	Initial release

Proprietary Notice

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

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 `rxpy` identifier indicates the revision status of the product described in this manual, for example, `r1p2`, where:

rx	Identifies the major revision of the product, for example, r1.
py	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™ Reference Systems Architecture Specification (CRSAS) Ma2.

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

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	Citations.
bold	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 underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

Convention	Use
`<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

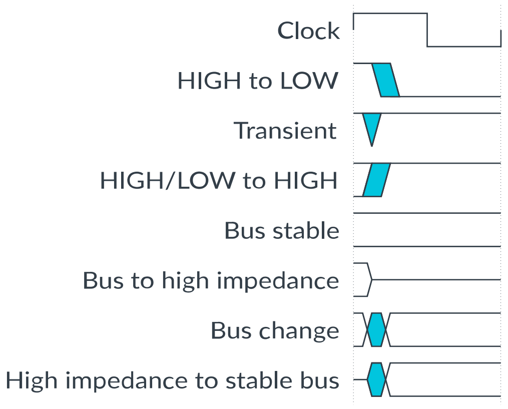
A reminder of something important that relates to the information you are reading.

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



Signals

The signal conventions are:

Signal level

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

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

Lowercase n

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

1.4 Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Arm® CoreLink™ DMA-350 Controller Technical Reference Manual	102482	Non-Confidential
Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual	101150	Non-Confidential
Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual	DDI 0571	Non-Confidential
Arm® CoreLink™ SIE-300 AXI5 System IP for Embedded Technical Reference Manual	101526	Non-Confidential

Arm product resources	Document ID	Confidentiality
Arm® CoreSight™ SDC-600 Secure Debug Channel Technical Reference Manual	101130	Non-Confidential
Arm® CoreSight™ System-on-Chip SoC-600M Technical Reference Manual	101883	Non-Confidential
Arm® Cortex®-M System Design Kit Technical Reference Manual	DDI 0479	Non-Confidential
Arm® Cortex®-M55 Processor Integration and Implementation Manual	101052	Confidential
Arm® Cortex®-M55 Processor Technical Reference Manual	101051	Non-Confidential
Arm® Cortex®-M85 Processor Integration and Implementation Manual	101925	Confidential
Arm® Cortex®-M85 Processor Technical Reference Manual	101924	Non-Confidential
Arm® Ethos™-U55 NPU Technical Reference Manual	102420	Non-Confidential
Arm® Ethos™-U65 NPU Technical Reference Manual	102023	Non-Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® AXI and ACE Protocol Specification	IHI 0022	Non-Confidential
AMBA® Low Power Interface Specification - Arm® Q-Channel and P-Channel Interfaces	IHI 0068	Non-Confidential
Arm® AMBA® 5 AHB Protocol Specification	IHI 0033	Non-Confidential
Arm® AMBA® APB Protocol Specification	IHI 0024	Non-Confidential
Arm® CoreSight™ Architecture Specification v3.0	IHI 0029	Non-Confidential
Arm® Debug Interface Architecture Specification ADIv6.0	IHI 0074	Non-Confidential
Arm® Key Management Unit (KMU) Specification	107715	Non-Confidential
Arm® Lifecycle Manager (LCM) Specification	107616	Non-Confidential
Arm® Platform Security Architecture, Trusted Base System Architecture for Arm®v6-M, Arm®v7-M and Arm®v8-M 2.0	DEN 0083	Confidential
Arm® Power Policy Unit Architecture Specification	DEN 0051	Non-Confidential
Arm® Security Alarm Manager (SAM) Specification	107716	Non-Confidential
Arm®v8-M Architecture Reference Manual	DDI 0553	Non-Confidential

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. Overview

The Corstone Reference Systems Architecture Specification (CRSAS) Ma2 specifies the architecture of a subsystem that integrates key components available from Arm which can be integrated into a larger system. This document describes a subsystem for Cortex-M processors that support MVE extensions. The subsystem forms the core of a full system. To form a full system, extra system peripherals must be added as expansion to the subsystem.

The subsystem integrates the following components:

- Cortex-M processor Core with *M-Profile Vector Extension* (MVE), FPU, DSP extensions, Caches, TCMs, and ETM. This revision of the CRSAS Ma2 supports Cortex-M85 or Cortex-M55 processors
- Ethos NPU core currently supports Ethos-U55 and Ethos-U65 neural processors
- Multiple banks of System Volatile Memory, for example SRAMs
- Memory Protection Controllers (MPC)
- Manager Security Controller (MSC)
- Exclusive Access Monitor (EAM)
- Key Management Unit (KMU)
- Lifecycle Manager (LCM)
- Security Alarm Manager (SAM)
- System interconnect
- Implementation Defined Attribution Unit (IDAU)
- CMSDK timers and watchdog timers
- Timestamp-based System timers and watchdog timers
- SDC-600 Secure Debug Channel
- Subsystem Controllers for security and general system control
- CoreSight SoC-600M components
- Power Policy Units (PPUs), Clock Controller and Low Power Interface interconnect components (PCK-600)

The CRSAS Ma2 specification does not include descriptions of other components that are not directly architecturally visible but are necessary to implement the system.

These components are integrated to implement the CRSAS Ma2 with the following features:

- TrustZone® aware system, with the system segregated into Secure and Non-secure worlds.
- Support for flexible banked volatile memory. This provides flexibility for the integrator and implementer to trade off between cost, complexity, memory throughput, and power consumption.
- Configurability to allow several features within the system to be included or removed.

- Power Control infrastructure, with several pre-defined voltage and power domains.
- Each switchable power domain has local power policy control, and coordinates with other power domains through a centralised dependency control and/or power interfaces. This provides the system with an autonomous dynamic power control infrastructure that, while being software configurable, aims to minimise software interaction.
- Clock control infrastructure that supports high-level clock control, including dynamic clock gating, and provides clock request handshakes to clock generators.
- Comprehensive reset generation and control.
- A CoreSight SoC based debug infrastructure that supports a shared Debug Access Port (DAP) and Trace Port, with support also for cross triggering.

The specification is implemented by the following Arm products:

- Corstone SSE-315

2.1 Document-specific conventions

In addition to the general conventions, there are document-specific conventions that apply.

Text

Text between < and > is a label to be replaced with the actual name or value of the item. For example, processor<n> where "n" is an instance number of either 0 or 1.

- Mathematical expressions are allowed within <> to offset the variable. For example, <NUMCPU +1>.
- <n> always denotes {0-<NUMCPU>}
- <m> always denotes {1-<NUMNPU-1>}

Text between { and } indicates a range to be replaced with the actual legal values. The allowed values can be:

- A range indicate by a start and end with a "-" or in-between. For example, {0-3} allows the any value between 0 and 3. One of the numbers can also be a variable. For example, {0-<NUMCPU>} where NUMCPU is a configuration value between 0 to 3.
- A list of discrete values separated by commas. For example, {0,1,2,3} allows the value to be any one of those values. One of the discrete values can also be a range. For example, {0, 3-6, 9} is the same as writing {0, 3, 4, 5, 6, 9}.
- A variable which has constraints. For example, <x> where x is {0-3} or <x> where x is between 0 and 3. Allows x to take any value between 0 and 3.

In the CRSAS Ma2 specification the terms 'subsystem implementer', and 'subsystem integrator' are used. The definition of these terms are:

Subsystem Implementer

Person implementing a subsystem according to this specification. This does not include the physical implementation of an existing frontend implementation of a subsystem implementation.

Subsystem Integrator

Person integrating this implementation of the subsystem, into a wider SoC, for example, by adding expansion components to the expansion ports.

Numbers

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

For both binary and hexadecimal numbers, where a bit is represented by the letter "x", the value is irrelevant. For example, a value expressed as 0b1x can be either `0b11` or `0b10`. To improve readability, long numbers can be written with an underscore separator between every four characters, for example, `0xFFFF_0000_0000_0000`.

Signals

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

- HIGH for active-HIGH
- LOW for active-LOW

A lowercase 'n' at the start or end of a signal name denotes an active-LOW signal.

The value of a multi-bit signal is defined using hexadecimal numbers.

When referring to bits within of a multi-bit signal, the <SIGNAL_NAME>[BIT_RANGE] formula is used. In the formula:

- SIGNAL_NAME is the name of the signal being referred to.
- BIT_RANGE is the bit range being referred to. If BIT_RANGE is omitted, then the text is referring to the whole signal.

For example, when referring to the bit range 31:2 of the Debug Access Interface address signal, the text reads:

- DEBUGPADDR[31:2]

Registers

When referring to registers and fields, the [<REGISTER_NAME>.<FIELD_NAME>[BIT_RANGE]] formula is used. In the formula:

- REGISTER_NAME is the short name of the register being referred to.
- FIELD_NAME is the name of the field within the register being referred to. If the FIELD_NAME is omitted, then the text is referring to the whole register.
- BIT_RANGE is the bit range, within the field, being referred to. If the BIT_RANGE is omitted, then the text is referring to the whole field or to the whole register if FIELD_NAME is also omitted. When referring to the bit ranges of a field, the field starts at 0.

For example, when referring to the bits 1:0 of the SYSVERSION register's DESIGNER_ID field, the text reads:

- SYSVERSION.DESIGNER_ID[1:0]

In the register bit field description tables, the following abbreviations are used to describe the accessibility of each bit field:

Table 2-1: Register bit field abbreviations

Abbreviation	Accessibility	Description
R	Read Accessible	Unless stated, these bit fields retain the value written to them until reset or powering down.
RW	Read and Write Accessible	Unless stated, these bit fields retain the value written to them until reset or powering down.
RO	Read only	These can only be read. Unless stated, any writes to these bit fields are ignored. This is similar to **WI**.
RAZ	Read as Zero	These always return Zeros for reads. Unless stated, writes proceed as normal.
RAO	Read as One	These always return Ones for reads. Unless stated, writes proceed as normal.
RAZ/WI	Read as Zero Write Ignores	These always return Zeros for read and ignores writes.
RAO/WI	Read as One Write Ignores	These always return Ones for read and ignores writes.
RAZ/WO	Read as Zero Write Only	These can only be written. These always return Zeros for reads.
W	Write Accessible	Unless stated, these bit fields retain the value written to them until reset or powering down.
WO	Write Only	These can only be written. Unless stated, any read from these bit fields returns zeros.
WI	Write Ignore	These ignore writes. Unless stated, reads proceed as normal. This is similar to RO.
W1S	Write 1 to Set	Each bit when written with one is set to one. Writing zeros to these are ignored.
W1C	Write 1 to Clear	Each bit when written with one is cleared to zero. Writing zeros to these are ignored.
W0S	Write 0 to Set	Each bit when written with zero is set to one. Writing ones to these are ignored.
W0C	Write 0 to Clear	Each bit when written with zero is cleared to zero. Writing ones to these are ignored.

Unless stated otherwise, after a register bit field is modified by a register access, it retains its values until a reset is applied or power is lost.

Where register values or behaviours are described as **IMPLEMENTATION DEFINED** (IMPL_DEF), the subsystem implementer has to assign the value or the behaviour. In certain cases the specification requires the value or behaviour to be selected from a list of options. Each implementation must specify the behaviour or register values as part of its own specification.

Where register values or behaviours are described as CONFIGURATION DEFINED (CFG_DEF), the value or behaviour depends on the configuration of the component.

Register values are defined by using actual values that are written in or read from the registers. They are expressed either as binary numbers or hexadecimal numbers. Alternatively, for single-bit values, they can be expressed as either 1 or 0, representing `0b1` and `0b0` respectively.

2.2 Compliance

The system architecture described in this document is designed to let you meet the requirements of Trusted Base System Architecture for Armv8-M.

2.2.1 Trusted Base System Architecture for Armv8-M

The *Arm® Platform Security Architecture, Trusted Base System Architecture for Arm®v6-M, Arm®v7-M and Arm®v8-M 2.0* (TBSA-M) is part of Arm Platform Security Architecture (PSA).

The TBSA-M implements best practice security principles when designing systems around Armv8-M Processing Elements (PEs). These principles support the design and integration of the following features rooted in hardware:

- Root of Trust (RoT)
- Protected key store
- Isolation between Secure and Non-secure software components
- Secure firmware update mechanism
- Lifecycle management mechanism for Secure control of debug, test, and access to provisioned secrets
- High-entropy random number generator for reliable cryptography
- Cryptographic acceleration

The CRSAS Ma2 specifies a system architecture that only partly fulfills the requirements specified within the TBSA-M. However, it specifies many features to help form the core of a system that complies.

In case of systems that integrate a CRSAS Ma2 based subsystem to comply with TBSA-M, the integrator is required to continue complying with TBSA-M requirements when expanding the system. For example:

- Suitable fuses have to be added to the LifeCycle Manager (LCM). While the LCM handles error detection, you must ensure that the fuses cannot be unprogrammed and that they are reliable and confidential.
- High-entropy random number generator and cryptographic implementation must be added.
- When adding more managers and subordinates to the system, the memory space must continue to obey Secure and Non-secure world partitioning.

Complying with TBSA-M requirements helps to create a Secure system, but it does not create a system that mitigates Denial of Service (DoS) or Side Channel Attacks (SCA). As such, DoS and SCA are out of scope of the CRSAS Ma2.

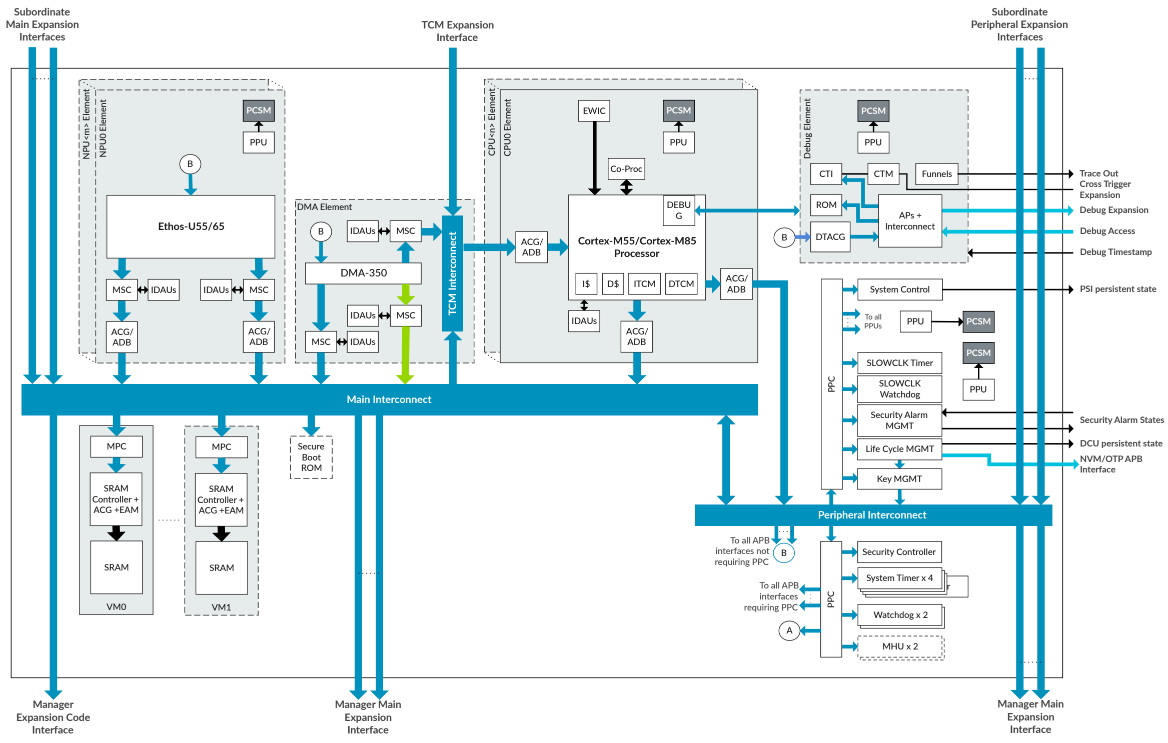
3. Topology

The CRSAS Ma2 topology partitions the design into functional blocks that are a combination of Arm IP and the supporting logic around them. The block-based design approach provides flexibility, scalability, and modularity.

For information about the functional blocks, see Functional block description.

For information about cross-block features, see Distributed functionality description.

Figure 3-1: Representative CRSAS Ma2 based system topology



The subsystem can be divided into the following functional blocks:

NPU block

Each NPU block contains an Ethos-U55 or Ethos-U65 NPU and its associated private infrastructure to integrate the NPU into the system. The CRSAS Ma2 supports zero to four NPU blocks within the system. For more information regarding the NPU, see NPU.

CPU block

Each CPU block contains an Armv8.1-M processor and its associated private infrastructure to integrate the processor into the system. The CRSAS Ma2 supports one to four CPU blocks within the system. For more information regarding the CPU, see CPU.

Main interconnect block

Connects all parts of the system to each other. For more information regarding the main interconnect, see System interconnect infrastructure.

Peripheral interconnect block

Provides access to lower performance and device type peripherals within the system. For more information regarding the peripheral interconnect, see System interconnect infrastructure.

DMA block

Provides DMA functionality, system, and expansion access to the processor TCMs. The TCM interconnect shown in the DMA block can be merged with the main interconnect as long as the connectivity shown is maintained. For more information regarding the DMA, see DMA.

Volatile memory bank block

Volatile memory banks collectively implement the main volatile storage within the subsystem, and implement the functionality to manage the volatile storage. For more information regarding the volatile memory, see Volatile memory. For more information regarding the volatile memory banks, see Volatile memory region.

Peripherals block

Defines a common set of peripherals expected in the system.

Security and system control block

Defines the infrastructure required to implement every configure, control, and monitor system states. These include security, clock, reset, and power control. For more information regarding the security and system control, see System and security control.

Debug system block

Defines the debug infrastructures that allow each processor and the system to be debugged securely. For more information regarding the debug system, see Debug Infrastructure.

For information about other components beyond the functional blocks, see the Distributed Functionality Description.

4. Functional block description

The following sections provides more details on the functionality of the various functional blocks.

4.1 NPU

The CRSAS Ma2 supports a configurable number of Ethos-U55 or Ethos-U65 NPU processors.

If implemented, each NPU can support a different static configuration, except for the CUSTOM_DMA_PRESENT configuration. If the CUSTOM_DMA_PRESENT configuration is supported by the NPU, it must be adhered to.

Table 4-1: NPU required static configuration

Configuration	Value for NPU<m>	Description
CUSTOM_DMA_PRESENT	0	It specifies whether the NPU includes an interface for a custom DMA: • 0: Interface not included • 1: Interface included

The Ethos-U55 or Ethos-U65 NPU do not support hardware debug halt control. We recommend that the debug software triggers a routine that controls the NPU when using hardware debug halt control of the processor and rest of the system for debugging.

The NPU can access the system memory map as follows:

- M0 Interface : All of the system memory except for the CPU<n> S-AHB Instruction TCM and CPU<n> S-AHB Data TCM areas
- M1 Interface : All of the system memory except for the CPU<n> S-AHB Instruction TCM and CPU<n> S-AHB Data TCM areas

The memory mapping is enforced by using:

- Manager Security Controllers (MSC), IDAUs, and MPCs for Secure and Non-secure world separation of memories.
- MSC, IDAU, and Peripheral Protection Controllers (PPC) for privileged and unprivileged separation and Secure and Non-secure world separation of peripherals.

Note

There is no protection provided for privileged or unprivileged memory access. If the risk of unprivileged software accessing privilege memory is acceptable, we recommend that the NPU is made available to unprivileged software.

For more information on Ethos-U55 or Ethos-U65, see *Arm® Ethos™-U55 NPU Technical Reference Manual* or *Arm® Ethos™-U65 NPU Technical Reference Manual*.

4.1.1 NPU security mapping

The NPU is capable of operating in different security modes. To change from operating in one security world to another, a reset is required.

When the PPU controlling the NPU power domain PD_NPU<m> releases the NPU reset, it samples the signals driven by:

- NPUSPPORSL.SP_NPU<m>PORSL to determine its default security level.
- NPUSPPORPL.SP_NPU<m>PORPL or NPUNSPORPL.NS_NPU<m>PORPL to determine the NPU's default privilege level.

For more information on the registers, see NPUSPPORPL and NPUNSPORPL. These registers are also used to control the PPC, which provides access protection to the NPU's configuration interface.

Note

The PPC protection setting takes effect as soon as the security and privileged registers are updated.

We recommend using the following sequences when transitioning the NPU to a new security or privilege level.

Changing security level from Secure (S) to Non-secure (NS)

The sequence details the steps for the Secure privilege software to change security level:

1. Read the current security from the corresponding system register: initial_security= NPUSPPORSL.SP_NPU<m>PORSL
2. If initial_security!=S, then end sequence. Otherwise software:
   a. Reads the privilege to be retained in the target security state from the corresponding system register: initial_privilege=NPUNSPORPL.NS_NPU<m>PORPL.
   b. Writes the current security alias of the NPU (S) on the configuration interface, targeting the CMD.power_q_enable register field of the NPU. This prevents the NPU powering OFF (0 = Power OFF denied) while preserving the rest of the CMD register.
   c. Writes the current security alias of the NPU (S) on the configuration interface, targeting the RESET.pending_CSL register field of the NPU. This sets the new security level (1= Non-secure) while preserving RESET.pending_CPL=initial_privilege.
   d. Reads repeatedly (poll) the current security alias of the NPU (S) from the configuration interface, targeting the STATUS register of the NPU. This continues until the field reset_status no longer returns the value 1.
   e. Writes the new security level into NPUSPPORSL.SP_NPU<m>PORSL register field (1 = Non-secure).
   f. Writes the current security alias of the NPU (NS) on the configuration interface, targeting the CMD.power_q_enable register field of the NPU. This allows the NPU to speculatively power OFF (1 = Power OFF allowed) while preserving the rest of the CMD register.

Changing security level from Non-secure (NS) to Secure (S)

The following sequence details the steps for the Non-secure privilege software to change security level to Secure:

1. Read the current security from the corresponding System register: initial_security=NPUSPPORSL.SP_NPU<m>PORSL
2. If initial_security!=NS, then end sequence. Otherwise software:
    a. Reads the privilege to be retained in the target Security state from the corresponding System register: initial_privilege=NPUSPPORPL.SP_NPU<m>PORPL.
    b. Writes the new security level into NPUSPPORSL.SP_NPU<m>PORSL register field (0 = Secure).
    c. Writes the current security alias of the NPU (S) on the configuration interface, targeting the CMD.power_q_enable register field of the NPU. This prevents the NPU powering OFF (0 = Power OFF denied) while preserving the rest of the CMD register.
    d. Writes the current security alias of the NPU (S) on the configuration interface, targeting the RESET.pending_CSL register field of the NPU. This sets the new security level (0=Secure) while preserving RESET.pending_CPL=initial_privilege.
    e. Reads repeatedly (poll) the current security alias of the NPU (S) from the configuration interface, targeting the STATUS register of the NPU. This continues until the field reset_status no longer returns the value 1.
    f. Writes the current security alias of the NPU (S) on the configuration interface, targeting the CMD.power_q_enable register field of the NPU. This allows the NPU to speculatively power OFF (1 = Power OFF allowed) while preserving the rest of the CMD register.

Changing privilege level

Secure or Non-secure privilege software can change the privilege state of the NPU as long as:

- The privilege state being moved to is equal or lower than the software enacting the change
- The security level of the software is at least matching the current security level (initial_security) of the NPU

The following sequence details the steps for the privilege software to change privilege level:

1. Read the current privilege level from the corresponding System register:

    Current security level = Non-secure

    initial_privilege=NPUNSPORPL.NS_NPU<m>PORPL

    Current security level = Secure

    initial_privilege=NPUSPPORPL.SP_NPU<m>PORPL
2. If initial_privilege is the desired one, then skip remaining sequence. Otherwise software:
    a. Writes the current security alias of the NPU (initial_security) on the configuration interface, targeting the CMD.power_q_enable register field of the NPU. This prevents the NPU from powering OFF (0 = Power OFF denied) while preserving the rest of the CMD register.
    b. Writes the current security alias of the NPU (initial_security) on the configuration interface, targeting the RESET.pending_CPL register field of the NPU. This sets the new privilege level (0=User; 1=privileged) while preserving RESET.pending_CSL=initial_security.

c. Reads repeatedly (poll) the current security alias of the NPU (initial_security) from the configuration interface, targeting the STATUS register of the NPU. This continues until the field reset_status no longer returns the value 1.

d. Writes the alias, corresponding to the current security level of the NPU (initial_security) of the new privilege level, into:

   Current security level = Non-secure

      NPUNSPORPL.NS_NPU<m>PORPL

   Current security level = Secure

      NPUSPPORPL.SP_NPU<m>PORPL

e. Writes the current security alias of the NPU (initial_security) on the configuration interface, targeting the CMD.power_q_enable register field of the NPU. This allows the NPU to speculatively power OFF (1 = Power OFF allowed) while preserving the rest of the CMD register.

4.2 CPU

The CRSAS Ma2 supports one to four ARMv8-M MVE processor cores.

Each processor can support different static configurations except the following that must be adhered to:

Table 4-2: Processor configurations that must be supported

Configuration	Value for CPU<n>	Description
SECEXT	1	Specifies whether the Armv8-M Security Extension is included: • 0: Security Extension not included. • 1: Security Extension included.
MPU_NS	> 0	Specifies the number of Non-secure MPU regions included.
MPU_S	> 0	Specifies the number of Secure MPU regions included.
SAU	> 0	Specifies the number of SAU regions included.
NUMIRQ	CPU<*n*>EXPNUMIRQ + 32	Specifies the number of external interrupts included for each processor.
IWIC	HASCPU<*n*>IWIC	Specifies whether the Internal Wake-up Interrupt Controller (IWIC) is included for each processor: • 0: IWIC not included • 1: IWIC included
WICLINES	CPU<n>EXPNUMIRQ + 35	Specifies the number of IRQ lines supported by the IWIC and EWIC. The value always includes the three internal events (NMI, RXEV, and Debug monitor event) and at least one IRQ.
CTI	If DEBUGLEVEL = 0 then 0, else 1	Specifies whether the Cross Trigger Interface (CTI) unit is included: • 0: CTI not included • 1: CTI included

Configuration	Value for CPU<n>	Description
BUSPROT	0	Specifies whether interface protection is supported on the M-AXI, P-AHB, EPPB, and S-AHB interfaces of the processor: • 0: Interface protection not included • 1: Interface protection included
LOCKSTEP	0	Specifies whether the processor is configured for dual-redundant lockstep operation. The options are: • 0: Regular processor operation • 1: Dual-redundant lockstep operation When LOCKSTEP is set to 1, we recommend that RAR is also set to 1. Otherwise, software must initialise all registers to prevent accidental triggering of the Dual-Core Lock-Step (DCLS) mismatch when the **UNKNOWN** state gets propagated to the top level. However, such software requirements might not be practical in certain software system environments. Therefore, the RAR option is strongly recommended for DCLS designs.
ITGU	1	Specifies whether the processor is configured with ITCM Security gate unit: • 0: TCM Gate Unit not included • 1: TCM Gate Unit included
DTGU	1	Specifies whether the processor is configured with DTCM Security gate unit: • 0: TCM Gate Unit not included • 1: TCM Gate Unit included
CFGMEMALIAS	`0b10000`	Memory address alias bit for the ITCM, DTCM, and P-AHB regions. The address bits used for the memory alias are: • `0b00001`: Alias bit = 24. • `0b00010`: Alias bit = 25. • `0b00100`: Alias bit = 26. • `0b01000`: Alias bit = 27. • `0b10000`: Alias bit = 28. • `0b00000`: No Alias. TCM security gating disabled.
CFGBIGEND	0	Data endian format: • 0: Little-endian (LE) • 1: Byte-invariant big-endian (BE8)
CFGITCMSZ	Must not be set to `0b0000`	Size of the instruction TCM region, encoded as: • = `0b0000`: No ITCM implemented, which is not supported by the CRSAS Ma2. • ≥ `0b0011`: $2^{CFGITCMSZ-1}$ KB. • Others: Reserved.
CFGDTCMSZ	Must not be set to `0b0000`	Size of the data TCM region, encoded as: • = `0b0000`: No DTCM implemented, which is not supported by the CRSAS Ma2. • ≥ `0b0011`: $2^{CFGDTCMSZ-1}$ KB. • Others: Reserved.

Configuration	Value for CPU<n>	Description
CFGPAHBSZ	Must be set to `0b010` - 128MB if Peripheral AHB (P-AHB) port exist.	Size of the Peripheral AHB (P-AHB) port memory region, encoded as: • = `0b000`: P-AHB disabled. • = `0b001`: 64MB • = `0b010`: 128MB • = `0b011`: 256MB • = `0b100`: 512MB When P-AHB port does not exist, this configuration is ignored.

Note

The processor configurations ITGUBLKSZ and DTGUBLKSZ are **IMPLEMENTATION DEFINED**. However, we recommend that ITGUBLKSZ and DTGUBLKSZ are equalised with VMMPCBLKSIZE.

Each processor also has several miscellaneous interface input signals that are tied, as shown in the following table.

Table 4-3: Processor interface ties that are required

Interface signals	Value for CPU<n>	Description
LOCKDCAIC	LOCKDCAIC	Disables access to the instruction cache and the direct cache access registers DCAICLR and DCAICRR in the processor.
INITTCMEN[1:0]	`0b11`	TCM enable initialization out of reset. Set all to '1' to enable both TCMs.
WICCONTROL[0]	1	Setting to '1' causes the processor to use WIC when in WFI DEEPSLEEP.
INITPAHBEN	1	Set to '1' to always enable P-AHB interface. This signal controls the reset value of PAHBCR.EN.

The WICCONTROL[0] signal is connected to bit [0] of the WICCONTROL[3:0] input signal of the Cortex-M55 or the Cortex-M85 processor.

4.2.1 EVENT interfaces

Each processor has event interfaces, the RXEV and TXEV signals. In the CRSAS Ma2 all the TXEV signals are logically connected and are used to drive the RXEV signal of the processors.

Note

In the CRSAS Ma2, events do not wake a processor from its EWIC based low power state or wake the system from the Hibernation{0/1} state.

The CRSAS Ma2 does not support the use of Wait For Event (WFE) for the processor to enter DeepSleep state.

4.2.2 Interrupts

The CRSAS Ma2 provides several events that can generate interrupts within the system.

The provided events are as follows:

- PPU interrupts
- Message Handling Units
- Security-based interrupts
- Timers and Watchdogs
- Cross Trigger interrupts
- NPUs
- DMA

In addition, depending on the configuration options, interrupts of each CPU<*n*> are made available to be driven through expansion logic. These interrupts are:

- CPU<*n*>EXPNUMIRQ
- CPU<*n*>EXPIRQDIS

Table 4-4: CPU <n> interrupt map on page 28 lists the interrupt map of each CPU<*n*>. For the first 32 interrupts, unless otherwise specified, all processor cores receive the same interrupt signals. Each CPU<*n*> only sees its own local CTIIRQ interrupts. Software must ensure that all the PPU interrupts are handled as Secure interrupts. If an interrupt source does not exist because of the chosen configuration of the system, the unused interrupt pin is not used. In these cases the interrupt is disabled and reserved.

Interrupts that also act as wakeup interrupts at each CPU<*n*>'s associated External Wakeup Interrupt Controller (EWIC) or Internal Wakeup Interrupt Controller (IWIC) are also indicated. The EWIC and IWIC act primarily as entities that take over the masking and holding of an interrupt. These entities act on behalf of the CPU<*n*>'s NVIC when the The CPU<*n*> is:

- In its OFF or low-power state
- Unable on its own to handle interrupts

When using the EWIC the system can:

1. Enter a lower power state that switches off each processor along with most of the system except the EWIC itself.
2. Run the EWIC on a much lower clock frequency to lower power consumption and attain a very low standby operating power.

An EWIC always exists with each CPU<*n*>, but the IWIC only exists for CPU<*n*> when HASCPU<*n*> IWIC = 1.

The following table shows interrupt inputs, sources for CPU<*n*>, and if they have WIC support.

Table 4-4: CPU <n> interrupt map

Interrupt input for CPU<*n*>	Interrupt Source for CPU<*n*>	WIC support
NMI	Combined Secure System Watchdog, SLOWCLK Watchdog, CPU<n>EXPNMI, and NMI of Security Alarm Manager. When LCM_KMU_SAM_PRESENT = 0, no NMI of the SAM is present. See Security Alarm Manager Response Action Settings.	Yes
IRQ[0]	Non-secure Watchdog Reset Request	Yes
IRQ[1]	Non-secure Watchdog Interrupt	Yes
IRQ[2]	SLOWCLK Timer	Yes
IRQ[3]	Timer 0	Yes
IRQ[4]	Timer 1	Yes
IRQ[5]	Timer 2	Yes
IRQ[6]	MHU 0 CPU<*n*> Interrupt. MHU interrupts are not shared and each processor only sees its own interrupt from the MHU. (Reserved if NUMCPU = 0). See Message Handling Unit.	Yes
IRQ[7]	MHU 1 CPU<*n*> Interrupt. MHU interrupts are not shared and each processor only sees its own interrupt from the MHU. (Reserved if NUMCPU = 0). See Message Handling Unit.	Yes
IRQ[8]	Reserved	-
IRQ[9]	MPC Combined (Secure)	Yes
IRQ[10]	PPC Combined (Secure)	Yes
IRQ[11]	MSC Combined (Secure)	Yes
IRQ[12]	Bridge Error Combined Interrupt (Secure)	Yes
IRQ[13]	Reserved	-
IRQ[14]	PPU_Combined	Yes
IRQ[15]	SDC-600. Reserved if SDC-600 does not exist	Yes
IRQ[16]	NPU0	Yes
IRQ[17]	NPU1	Yes
IRQ[18]	NPU2	Yes
IRQ[19]	NPU3	Yes
IRQ[20]	Key Management Unit When LCM_KMU_SAM_PRESENT = 0 this IRQ is Reserved	Yes
IRQ[23:21]	Reserved	-
IRQ[24]	DMA (Combined Secure)	Yes
IRQ[25]	DMA (Combined Non-secure)	Yes
IRQ[26]	DMA (Security violation)	Yes
IRQ[27]	Timer 3 AON	Yes
IRQ[28]	CPU<n>CTIIRQ0 (local CPU CTI only)	No
IRQ[29]	CPU<n>CTIIRQ1 (local CPU CTI only)	No
IRQ[30]	Critical Severity Fault Interrupt of Security Alarm Manager. When LCM_KMU_SAM_PRESENT = 0 this IRQ is Reserved. See Security Alarm Manager Response Action Settings.	Yes

Interrupt input for CPU<*n*>	Interrupt Source for CPU<*n*>	WIC support
IRQ[31]	Severity Fault Interrupt of Security Alarm Manager. When LCM_KMU_SAM_PRESENT = 0 this IRQ is Reserved. See Security Alarm Manager Response Action Settings.	Yes
IRQ[<CPU<n>EXPNUMIRQ +31>:32]	CPU<n>EXPIRQ[CPU<n>EXPNUMIRQ-1:0]. See Interrupt Interfaces.	Yes

4.2.3 Processor Custom Datapath Extension interface

If supported by the processor core, each processor core of the subsystem can be configured to have a Custom Datapath Extension interface. The method to determine if this interface is present is **IMPLEMENTATION DEFINED**.

If a CPU<*n*> Custom Datapath Extension interface exists, then the protocol of this interface is **IMPLEMENTATION DEFINED**.

These interfaces reside in the PD_CPU<n> power domain of their respective processors:

- CPUCPU<n>CLK clock domain
- nWARMRESETCPU<n> reset domain.

Note

Depending on the actual processor implementation, the reset can be a special output that is dependent on at least the nWARMRESETCPU<n>.

For more information on the Custom Datapath Extensions and related interfaces, see *Arm® Cortex®-M55 Processor Integration and Implementation Manual* or *Arm® Cortex®-M85 Processor Integration and Implementation Manual*.

4.3 System interconnect infrastructure

The system interconnect infrastructure provides a bus infrastructure that transfers memory mapped access from bus managers to subordinates in the system. The CRSAS Ma2 defines two key interconnects that form the system interconnect and they are:

Main interconnect

This interconnect is expected to provide the highest amount of bus throughput. It is primarily but not exclusively, for code and data accesses that targets memories or high throughput interfaces. For example:

- In-subsystem volatile memories
- Flash, DRAM controllers or ROM that reside outside the system
- Other high throughput devices

While the performance and protocol of the interconnect is **IMPLEMENTATION DEFINED**, we recommend that this interconnect is implemented to match the throughput capability of the processor. The main interconnect provides access support for all memory regions that are not private to the processors. However, access to the following regions are forwarded to the peripheral interconnect:

- `0x4000_0000` to `0x5FFF_FFFF`.
- `0xE010_0000` to `0xFFFF_FFFF`.

Peripheral interconnect

This interconnect is expected to provide access to lower performance peripherals. For example:

- System and Watchdog Timers
- System and other security configuration registers
- Power control logic

A Cortex-M MVE based processor has direct interface to access this nterconnect. Any access that does not reside in the following region must be routed to the Main interconnect:

- `0x4000_0000` to `0x5FFF_FFFF`
- `0xE010_0000` to `0xFFFF_FFFF`

While the performance and protocol of the interconnect is **IMPLEMENTATION DEFINED**, we recommend that this interconnect is implemented to have a lower throughput performance compared to the main interconnect. We also recommend that this interconnect supports a bus protocol that matches those needed by the peripherals. The latency of the interface is also expected to be reduced.

Both interconnects must be able to achieve the following:

- Transfer the several key properties of the access, from a manager to a subordinate. This is required to ensure that any security-related infrastructure can have all the necessary information to make decisions on access rights. The key properties are as follows:
  - Security attribute indicating whether an access is marked as Secure or Non-secure.
  - The privilege level of the access. The level is either privileged or unprivileged, or of a similar type.
  - Bus access address, read/write attribute, and (optionally) manager identity.
- Complete an access that has been issued. At completion of the access, minimally communicate whether the access is successful or has failed.
- Support the ability to automically read and modify a memory location. For example, through the implementation of read lock and write conditional type exclusive memory access.

The Interconnect resides across PD_SYS, potentially across PD_AON, and even PD_MGMT if it exists. The interconnect runs primarily on SYSCLK.

TCM Interconnect

This interconnect provides access to the Tightly Coupled Memories (TCM) that are internal to the CPU<n> from:

- TCM subordinate interface
- Main interconnect
- The DMA

The TCM interconnects follow the mapping and properties of the TCM subordinate interface.

While this architecture defines these three interconnects as seperate, a user can choose to implement them as one or two merged interconnects. This is permissible as long as the software view of the system are not altered.

4.3.1 ACC_WAIT control

The CRSAS Ma2 provides a set of controls to allow user to add access control gates or block access to the system expansion interfaces.

ACC_WAIT control allows user to:

- Add access control gates that are driven by using the ACCWAITn signal
- Block access to the system, when:
  - System leaves HIBERNATION{0-1} state
  - System resets
  - System performs its first power up
  - Software wants to reconfigure security settings in the system

  ACC_WAIT control prevents access to the system until all security-related features of the system are set up correctly. After software is ready, it can release the gates by writing to the BUSWAIT.ACC_WAITN register. Then, software can check the current status of all external gating units by reading the ACCWAITNSTATUS signal value through the BUSWAIT.ACC_WAITN_STATUS register.

  For more information about the register and its relevant fields, see BUSWAIT.

  For more information about the ACCWAITNSTATUS signal, see Other Security Expansion signals.

4.4 DMA

The CRSAS Ma2 supports an optional DMA-350.

NUMDMA represents the number of DMA cores present in the system:

- NUMDMA = 0, the DMA IP and its associated integration logic does not exist within the system.

- NUMDMA = 1 the DMA IP and its associated integration logic does exist within the system.

For information about NUMDMA, see Common configuration options.

When NUMDMA = 1, the DMA can support any static configuration. However, the following must be adhered to:

Table 4-5: DMA required static configurations

Configuration	Value for DMA	Description
AXI5_M1_PRESENT	1 when using DMA-350	Specifies whether an additional manager port is present. When set, the m1 manager port is present • 0: Interface not included • 1: Interface included
SECEXT_PRESENT	1	Specifies where TrustZone security is supported • 0: Security Extensions are not implemented • 1: Security Extensions are implemented
NUM_CHANNELS	16 >= NUM_CHANNELS >= (NUMCPU+1)*2	Specifies the number of DMA channels.

We recommend that:

- The DMA channels are not shared between security and privilege levels
- Each security and privilege level that requires DMA support has a dedicated channel configured in the DMA

The DMA can access the system memory map as follows:

M0 interface

All of the system memory. However, when using DMA-350, the CPU<n> S-AHB Instruction TCM and CPU<n> S-AHB Data TCM areas are not accessible from this interface.

M1 interface

Only the CPU<n> S-AHB Instruction TCM and CPU<n> S-AHB Data TCM areas.

Security mapping is enforced using:

- Manager Security Controllers (MSC), IDAUs, and MPCs for Secure and Non-secure world separation of memories
- MSC, IDAU, and PPC for privileged and unprivileged separation and Secure and Non-secure world separation of peripherals

Note

There is no protection provided for privileged or unprivileged memory access. Therefore, we recommend that only privileged software is allowed to program the DMA.

The DMA can be configured with 0-32 trigger in and trigger out interfaces. For more information on the DMA configuration, see *Arm® CoreLink™ DMA-350 Controller Technical Reference Manual* .

Other configuration parameters may also create DMA interfaces, such as GPO, Stream interfaces, and others. These interfaces are routed directly to the top level of the subsystem and are in the same power and clock domain as the DMA.

Both DMAs support automatic booting capability. This allows the DMA to optionally run a sequence of DMA operation after reset without software setup. We recommend that this functionality is not used and the boot_En signal of the DMA is tied LOW. However, if you choose to utilise the automatic booting capability, you must ensure that the DMA operations performed are in line with the security requirements for your system. In addition, the boot_En signal of the DMA should also be forced LOW when LCMRSTREQ is asserted. This is a measure to block the DMAs from executing any commands after DMA reset and during the provisioning of confidential assets using the LCM Secure Provisioning feature.

4.5 Volatile memory

The CRSAS Ma2 can support 0 to 4 Volatile Memory (VM) banks, VM<x>. The number of memory banks is defined by the NUMVMBANK configuration, and therefore *x* is 0 to NUMVMBANK-1. When when NUMVMBANK = 0 there is no VM bank.

Each VM can support a configurable amount of SRAM memory, but must obey the following:

- The size of each is powers of two
- The size of each bank is defined using the `VMADDRWIDTH` configuration option and is equal to $2^{\text{VMADDRWIDTH}}$ bytes
- The total memory size of all VM banks that exist within the system combined is less than 16Mbytes
- All VM forms a contiguous area of memory and is mapped to a starting address of `0x2100_0000`, which is also aliased to a starting address of `0x3100_0000`

All VM<x> must support exclusive accesses from the processors and external managers.

Each VM<x> has a Memory Protection Controller (MPC) associated with it that provides the ability to map segments of each memory to Secure world or Non-secure world. For more information on MPC, see *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual* and *Arm® CoreLink™ SIE-300 AXI5 System IP for Embedded Technical Reference Manual*.

The CRSAS Ma2 requires that all VMs use the same MPC block size as defined by the configuration VMMPCBLKSIZE. In addition, the cfg_init_value input pin of each MPC, which defines the security configuration of the memory regions at reset, must be set to `0b0`. In this case, out of reset, all memory locations are mapped to the Secure world by default.

The CRSAS Ma2 supports a power domain for each bank, PD_VMR<x>. Each power domain only contains the actual VM of the memory bank with all other logic of the memory banks, including the MPC, residing in PD_SYS. Therefore any power gating or retention only refers to the memory itself. For more information on power control of PD_VMR<x> see BR_SYS power modes.

All VM<x>s run on SYSSYSCLK.

For more information about the `NUMVMBANK` and `VMADDRWIDTH` configurations, see Main system configuration options.

For more information about the SYSSYSCLK, see Input and output clocks.

4.6 System and security control

The CRSAS Ma2 provides several registers to allow various features of the system to be discovered, configured, and controlled. These registers are grouped into four register blocks:

Secure Access Configuration register block

This register block provides registers to configure PPC and Manager Security Controllers (MSC) that reside in the system and in the expansion system. The registers configure the PPC and MSC through the Security Control Expansion interface. These are Secure access-only registers. For more information on the registers implemented in this block, see Secure Access Configuration register block.

Non-secure Access Configuration register block

This register block provides registers to configure PPC that reside in the system and in the expansion system. The registers configure the PPC through the Security Control Expansion interface. These are Non-secure access-only registers. For more information on the registers implemented in this block, see Non-secure Access Configuration register block.

System Information register block

This register block provides information on the system configuration and identity. For more information on the registers implemented in this block, see SYSINFO register block.

System Control register block

This register block provides registers for power, clocks, resets, and other general system control. For more information on the registers implemented in this block, see System Control Register block.

4.7 Debug infrastructure

The CRSAS Ma2 supports two possible configurations that define the extent of its debug system infrastructure.

The configurations defining the extent of the debug system infrastructure are as follows:

HASCSS = 0

In this basic debug configuration a CoreSight SoC-600M based common debug infrastructure is not defined and does not exist. Instead, all debug interfaces are brought out from the processor as expansion interfaces. When HASCSS = 0, NUMCPU must be 0.

HASCSS = 1

In this full debug configuration, a CoreSight Soc-600M based common debug infrastructure exists and is called the Debug System.

A debug access to any of the systems obeys normal memory map and decoding rules. A Debug access must not cause a security violation interrupt to be generated, even if the debugger performs an operation that would normally cause one.

4.7.1 Basic debug configuration

When HASCSS = 0, there can only be one processor, and the system does not include a common shared CoreSight SoC-600M based debug infrastructure.

Without the common shared CoreSight SoC-600M based debug infrastructure, the following applies:

- The separate debug power domain of the processor, if it exists, is merged into a single power domain, the PD_DEBUG. The power domain is controlled using a single PPU, the DEBUG_PPU.
- The debug interfaces of the processor, if they exist depending on the DEBUGLEVEL configuration, are brought out as expansion interfaces. See Debug and trace related interfaces.

4.7.2 Full debug configuration

When HASCSS = 1 and DEBUGLEVEL > 0, the system contains a CoreSight SoC-600M based debug infrastructure.

CoreSight SoC-600M based debug infrastructure provides the following functionality:

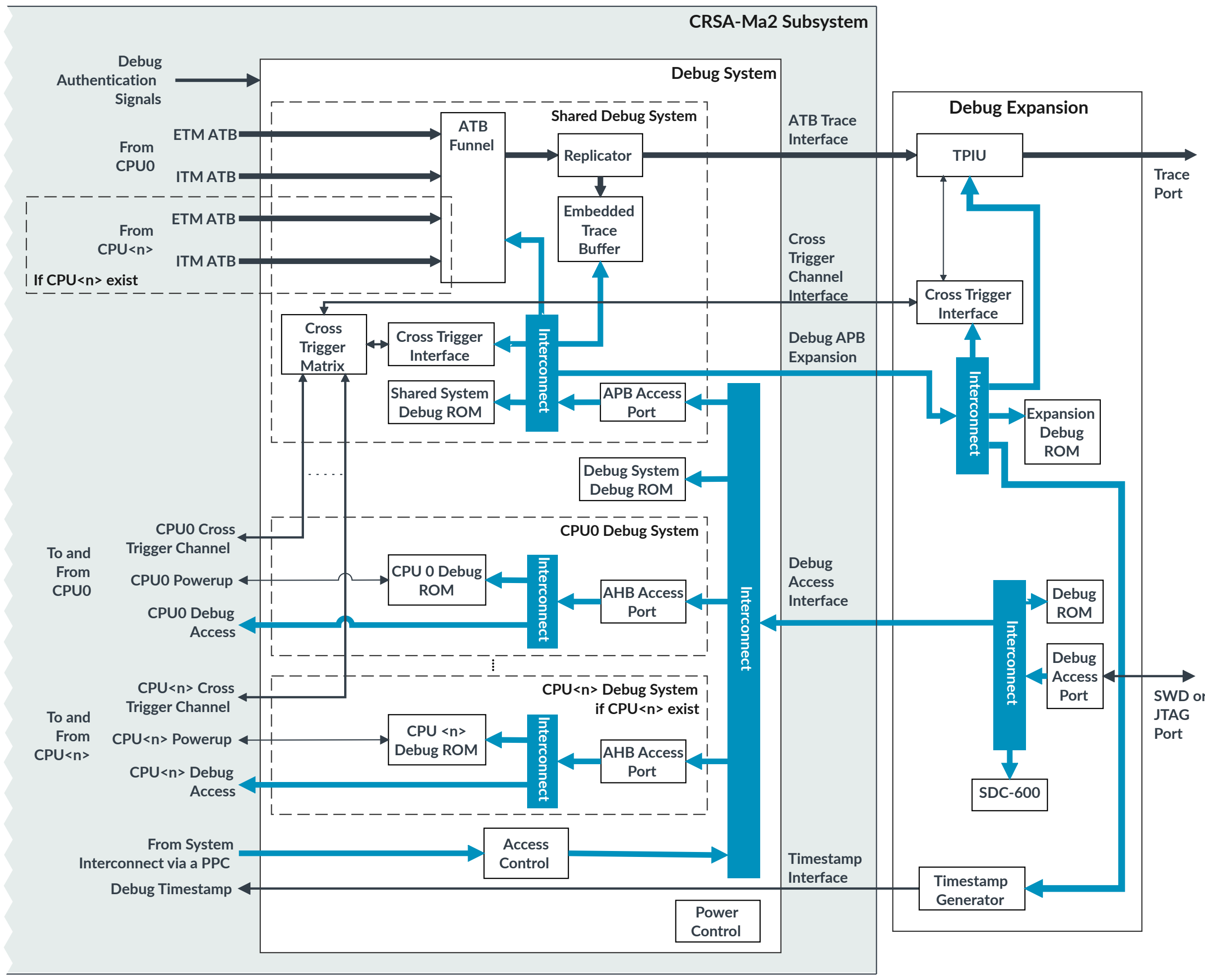
- Single CoreSight DAP to access a number of Coresight Access Ports (APs). Each of these next level Coresight APs then provide debug access to each individual processor core and to the Shared Debug System that they all share.
- When DEBUGLEVEL > 1, a shared trace infrastructure where all trace data are funneled into a single trace data output, or to an Embedded Trace Buffer (ETB).
- System subordinate interface to the main interconnect to provide system access to the CoreSight APs.
- Cross Trigger Matrix (CTM) and Cross Trigger interfaces to distribute triggers between the processors and the Shared Debug System.
- Expansion interfaces which allow additional CoreSight components to be added according to the specific needs of an SoC.

Figure 4-1: Example debug subsystem structure on page 36 shows a representative example architecture block diagram of the Debug System.

Note

The example does not show **IMPLEMENTATION DEFINED** components, or blocks that may be required because of clock, reset, and power domain crossing. The example also does not include bus width conversion, or buffering that is not directly visible to software.

Figure 4-1: Example debug subsystem structure



In Figure 4-1: Example debug subsystem structure on page 36, the block "Debug System" is shown as composed of a number of lower level systems blocks that it provides access to. These blocks include:

- CPU<n> Debug Systems, which are each associated with a processor in the system.
- Shared Debug System, which all processors share to provide the ability to merge trace data, support cross triggering, and to support debug expansion.

4.7.2.1 Debug Access

The CoreSight SoC-600M based debug infrastructure provides a single APB4 Debug Access interface for an expansion Debug Access Port (DAP) to connect to a minimum of two and up to five Memory Access Ports (MEM-AP), along with a Debug ROM table.

The purpose of the MEM-AP and the Debug ROM table are as follows:

- One MEM-AP is used for accessing the Shared Debug System infrastructure components, including the debug expansion logic on the Debug APB Expansion interface.
- One MEM-AP is used for each processor in the system that provides access to each CPU<n> Debug System's access interface.
- The Debug System ROM table provides information about the MEM-APs.

For debug via the APB4 Debug Access interface, accesses out of the above MEM_APs are control by Debug Authentication signals. DAPDSSACCEN provides overall access control, combined with DBGEN or NIDEN to control Non-secure access, and SPIDEN or SPNIDEN to control Secure access.

System interconnect access to the MEM-APs is also provided. Access is gated to control which of the processors in the system, or **IMPLEMENTATION DEFINED** manager, or group of managers are allowed to access the MEM-APs components. Access gating is done by using the SYSDSSACCEN<n> and SYSDSSACCENX Debug Authentication signals. Once access is granted to the debug system, DBGEN or NIDEN then controls Non-secure access and SPIDEN or SPNIDEN controls Secure access.

Access via the system interconnect are mapped to the addresses:

- from `0xE010_0000` to `0xE01F_FFFF` in the Non-secure world
- from `0xF010_0000` to `0xF01F_FFFF` in the Secure world

The PERIPHNSPPC0.NS_SYSDSS register determines which of the two regions is accessible and the PERIPHSPPPC0.SP_SYSDSS determines the privilege level.

CPU debug access

To provide debug access to each processor core, the CRSAS Ma2 provides a MEM-AP accessible through the Debug Access interface. From this MEM-AP, a Class `0x9` Debug ROM table is provided to:

- Point to the CPU<n> ROM table that is private to each processor at PPB address region at `0xE00F_F000`. This ROM table also provides Granular Power Requestor (GPR) function to allow the debugger to wake CPU<n>.
- Optionally, also point to another ROM table: CPU<n> MCU ROM table. This ROM table is added by the system integrator that resides in the PPB address region and in the EPPB expansion bus at the address: CPU<n>MCUROMADDR. The existence of the pointer to the CPU<n> MCU ROM is CFG_DEF.

All access from the MEM-AP that does not access the ROM table is to be forwarded to the Debug Access interface of the processor. It is **IMPLEMENTATION DEFINED** if the access goes through power and/or clock crossing bridge.

Note

Accessibility through the Debug Access interface of the processor depends on the DBGEN and SPIDEN Debug Authentication signals, and the DAUTHCTRL.UIDEN register in the processor core. For more information, see *Arm® Cortex®-M55*

Processor Technical Reference Manual or *Arm® Cortex®-M85 Processor Technical Reference Manual*.

Shared Debug System Access

The CRSAS Ma2 provides a MEM-AP that is accessible through the Debug Access interface to provide access to the Shared Debug System. This Shared Debug System provides the following:

- Shared Debug System CoreSight ROM that describes all debug components in the Shared Debug System accessible through this MEM-AP. This ROM table includes an entry pointing to an external CoreSight ROM table at the `0x0008_0000` address through the Debug APB Expansion interface, which defines what the system integrator has added as expansion.
- Trace Funnel to funnel all trace data sources together.
- Replicator and an Embedded Trace Buffer (ETB) to allow traced data to be either:
    - To be forwarded to a Trace Port Interface Unit (TPIU) in the expansion logic
    - To be temporarily stored in the ETB so that the software or the debugger can read them through the Debug Access interface
- CTM and CTI that support triggers to and from:
    - DMA, Timers, and Watchdogs in the system
    - Triggers from the processor cores
    - Triggers from the expansion system
- Debug APB Expansion interface to add debug components to the system. This allows a system integrator to extend the Shared Debug System in accordance with the needs of the SoC.

For example, in Full Debug Configuration the Debug Expansion adds the following to complete the debug solution as a standalone microcontroller:

- DAP to allow an external debugger to access the platform.
- Access Control Gate, controlled using DAPDSSACCEN Debug Authentication Access Control signal.
- CTI to provide triggers to and from the Trace Port Interface Unit (TPIU).
- TPIU to output trace data.
- SDC-600 Secure Debug Channel External APBCOM to provide an interface for Secure communications between the debugger and the Secure firmware in the system. Through the interface Secure debug certificates can be injected into the platform. The External APBCOM connects to the optional SDC-600 Secure Debug Channel Internal APBCOM. For more information, see Secure Debug Channel registers.
- Debug Timestamp generator.

4.7.2.2 Timestamps

The CRSAS Ma2 provides a debug timestamp input, TSVALUE<B/G>[63:0], which is used by all debug logic that requires timestamp within the subsystem. These are primarily the processor cores in the system.

Since each processor core can be running on different clock rates, the timestamp input updates might occur at a rate that is slower than some of these clocks. The CRSAS Ma2 does not specify the inclusion of timestamp interpolators to be deployed in the system for faster cores. As a result, when the processor clock to the timestamp update rate ratio increases, especially beyond 10:1, it degrades the timestamp granularity. This reduces the ability for you to deduce traced event timings accurately between the two clock domains.

4.7.2.3 Cross Trigger

The Shared Debug System implements a Cross Trigger Matrix (CTM) and a single Cross Trigger Interface (CTI). These together allow the DMA, the timers, and the watchdog timers in the CRSAS Ma2 subsystem to be halted and restarted. The halting and restarting occurs by using trigger sources from any of the processor cores and also from trigger sources external to the subsystem through the Cross Trigger Channel interface.

The first four Cross Trigger inputs and the first six outputs are reserved for internal use to support the following:

- SLOWCLK watchdog and SLOWCLK timer halting as follows:
  - CTIEVENTOUT[0] of the CTI is used internally to halt the SLOWCLK timer and watchdog.
  - CTIEVENTOUT[1] of the CTI is used internally to restart the SLOWCLK timer and watchdog.
- Triggers to and from the ETB:
  - CTIEVENTOUT[2] of the CTI is used to drive TRIGIN input of the ETB to request the ETB to insert a trigger in a trace stream.
  - CTIEVENTOUT[3] of the CTI is used to drive FLUSHIN input of the ETB to initiate a flush.
  - CTIEVENTIN[0] of the CTI takes the event output FLUSHCOMP of the ETB to indicate that a flush operation is completed.
  - CTIEVENTIN[1] of the CTI takes the event output ACQCOMP of the ETB to indicate that a trace acquisition is completed.
  - CTIEVENTIN[2] of the CTI takes the event output FULL of the ETB to indicate that either the trace memory is full or the write pointer wrapped around.
  - CTIEVENTIN[3] of the CTI is reserved and tied 0.
- If NUMDMA > 0, triggers a halt and restart the DMA, reserved if NUMDMA=0
  - CTIEVENTOUT[4] of the CTI is used internally to halt the DMA.
  - CTIEVENTOUT[5] of the CTI is used internally to restart the DMA.

All other CTI inputs and outputs of the CTI block are made available as expansion signals through CTIEVENTIN[<NUM_EVENT_SLAVES-1>:4] and CTIEVENTOUT[<NUM_EVENT_MASTERS-1>:6].

NUM_EVENT_SLAVES and NUM_EVENT_MASTERS are defined by the NUM_EVENT_SLAVES and NUM_EVENT_MASTERS configuration of the css600_cti device respectively. For the CRSAS Ma2, both NUM_TRIG NUM_EVENT_SLAVES and NUM_EVENT_MASTERS must be >= 8.

For more information on the css600_cti, see *Arm® CoreSight™ System-on-Chip SoC-600M Technical Reference Manual*.

When integrating a CRSAS Ma2 based subsystem with HASCSS=1, we recommend that the trigger signals are used to also halt the System Timestamp counter in the following way:

- Use CTIEVENTOUT[6] to halt the System Timestamp counter.
- Use CTIEVENTOUT[7] to restart the System Timestamp counter.

Note

It is the responsibility of the debug software to halt and restart NPUs in the system, using their programming interface.

4.7.2.4 Trace Infrastructure

When DEBUGLEVEL = 2, the CRSAS Ma2 provides a trace infrastructure that funnels all trace source from all processor cores into a single trace data stream.

The single trace date stream is replicated. One of the streams drives the ATB Trace interface that is expected to be picked up by a TPIU. The other stream is taken up by the ETB, allowing the trace data to be read by the software or by an external debugger through the Debug Access interface.

When DEBUGLEVEL < 2, the CRSAS Ma2 does not provide any trace infrastructure.

5. Distributed Functionality Description

Distributed functionalities are components that are beyond the functional blocks of the system.

The distributed functionality components are as follows:

- Clocking infrastructure
- Reset infrastructure
- Read-only memory
- Timers and watchdogs
- Message Handling Unit
- Power Policy Units
- Peripheral Protection Controllers
- Memory Protection Controllers
- Manager Security Controllers
- Lifecycle Manager
- Key Management Unit
- Security Alarm Manager
- Power control infrastructure

5.1 Clocking infrastructure

The CRSAS Ma2 provides several input clocks to the system.

Available input clocks of the CRSAS Ma2 are as follows:f

SLOWCLK

An always running clock that is expected to be the first available clock when the system first powers up.

This clock is required to run even when others can be turned off in the lowest power state of the system other than OFF.

This clock minimally drives the following logic that resides in the PD_AON power domain:

- 32-bit timer: This timer, running at SLOWCLK, allows the user to set a wake event.
- 32-bit watchdog timer. This watchdog timer provides protection against an unresponsive system, particularly during the lowest power state.

AONCLK

This clock drives all other logic that resides in the PD_AON domain. This does not include logic in the PD_MGMT power domain that is merged into PD_AON when PILEVEL < 2.

The Q-Channel Control interface (AONCLK Q-Channel Control interface) allows the subsystem to request and handshake the availability of the AONCLK.

This clock drives the following:

- External Wakeup Interrupt Controller<n> (EWIC<n>) that allows a processor to be woken through an interrupt.
- The PD_MGMT PPU.
- All other logic in the PD_AON domain that is not running on SLOWCLK or SYSCLK when PILEVEL < 2.

For more information on the various PILEVEL, see Power control infrastructure.

SYSCLK

This is the main system clock that drives most of the system that resides in the PD_SYS power domain.

This clock also drives all logic that resides in the PD_MGMT domain. The Q-Channel Control interface (SYSCLK Q-Channel Control interface) allows the subsystem to request and handshake the availability of the SYSCLK.

This clock is requested to run in any of the following cases:

- System is not in HIBERNATION{0-1} or in SYS_RET power state.
- Clock is forced to run through the CLOCK_FORCE register.
- Activity based clock request from the logic associated with this clock.

This clock drives the following:

- Main interconnect and peripheral interconnect and other related functionality like expansion interfaces.
- System and Security Control related registers and logic, except those that reside in PD_AON.
- Power Control logic that resides in the PD_MGMT power domain and controls the PD_SYS, PD_CPU<n>, PD_DEBUG, and PD_NPU<m> power domains.
- All volatile memory interfaces and peripherals in the PD_SYS domain, along with the interfaces of timers and watchdog timers.

CPU<n>CLK

Each clock input drives a single processor core and its associated expansion interfaces and integration logic. The Q-Channel Control interface (CPU<n>CLK Q-Channel Control interface) allows the subsystem to request and handshake the availability of the CPU<n>CLK.

Each clock is expected to be requested to run in any of the following cases:

- Its associated processor core domain (PD_CPU<n>) is not in OFF, MEM_RET, *or FULL_RET* state.
- The PD_DEBUG power domain is not in OFF state if the processor has constraints tying the debug power domain of the processor core to its debug logic.

- It is forced to run through the CLOCK_FORCE register.
- It receives a clock request from the logic associated with this clock.

When PILEVEL=0, the PD_CPU0 and PD_DEBUG are merged into PD_SYS.

DEBUGCLK

This clock must exist when HASCSS = 1.

This clock drives the Debug System. The Q-Channel Control interface, (DEBUGCLK Q-Channel Control interface) allows the subsystem to request and handshake the availability of the DEBUGCLK.

This clock is expected to be requested to run in any of the following cases:

- The Debug System, PD_DEBUG, is not in OFF state.
- It is forced to run through the CLOCK_FORCE register.
- It receives a clock request from the logic associated with this clock.

When PILEVEL=0, PD_DEBUG are merged into PD_SYS.

CNTCLK

This clock is associated with the System Timestamp input, CNTVALUE<G/B>.

This clock is used to drive timestamp related logic in all timestamp-based timers and watchdogs.

NPU<m>CLK

These clocks must be present when NUMNPU > 0. Each clock input drives a single NPU core and its associated interfaces and integration logic. The Q-Channel Control interface (NPU<*m*>CLK Q-Channel Control interface) allows the subsystem to request and handshake the availability of the NPU<m>CLK.

Each clock is expected to be requested to run in any of the following cases:

- Its associated NPU domain, PD_NPU<m>, is not in OFF state.
- It is forced to run through the CLOCK_FORCE register.
- It receives a clock request from logic associated with this clock.

The granularity of switching each clock as controlled by their respective clock Q-channel control interface is **IMPLEMENTATION DEFINED**. We recommend that all Q-channel control interfaces defined here are used in relation to the system power state, where each clock is requested to turn on or off depending on the power state of the domains that use them. The expectation is that there is a higher cycle cost in starting and stopping these clock sources. We do not recommend to use these clock Q-channel control interfaces to perform activity based high-level clock gating. Instead, an implementation of the CRSAS Ma2 should perform activity based high-level clock gating internally.

Figure 5-1: Clock tree example when PILEVEL=2 on page 44 shows an example cock that illustrates how the output clocks are related to the input clocks and their power domain relationships, when PILEVEL=2.

The figure does not show internal Q-channel and P-channel infrastructure that is required for each clock controller.

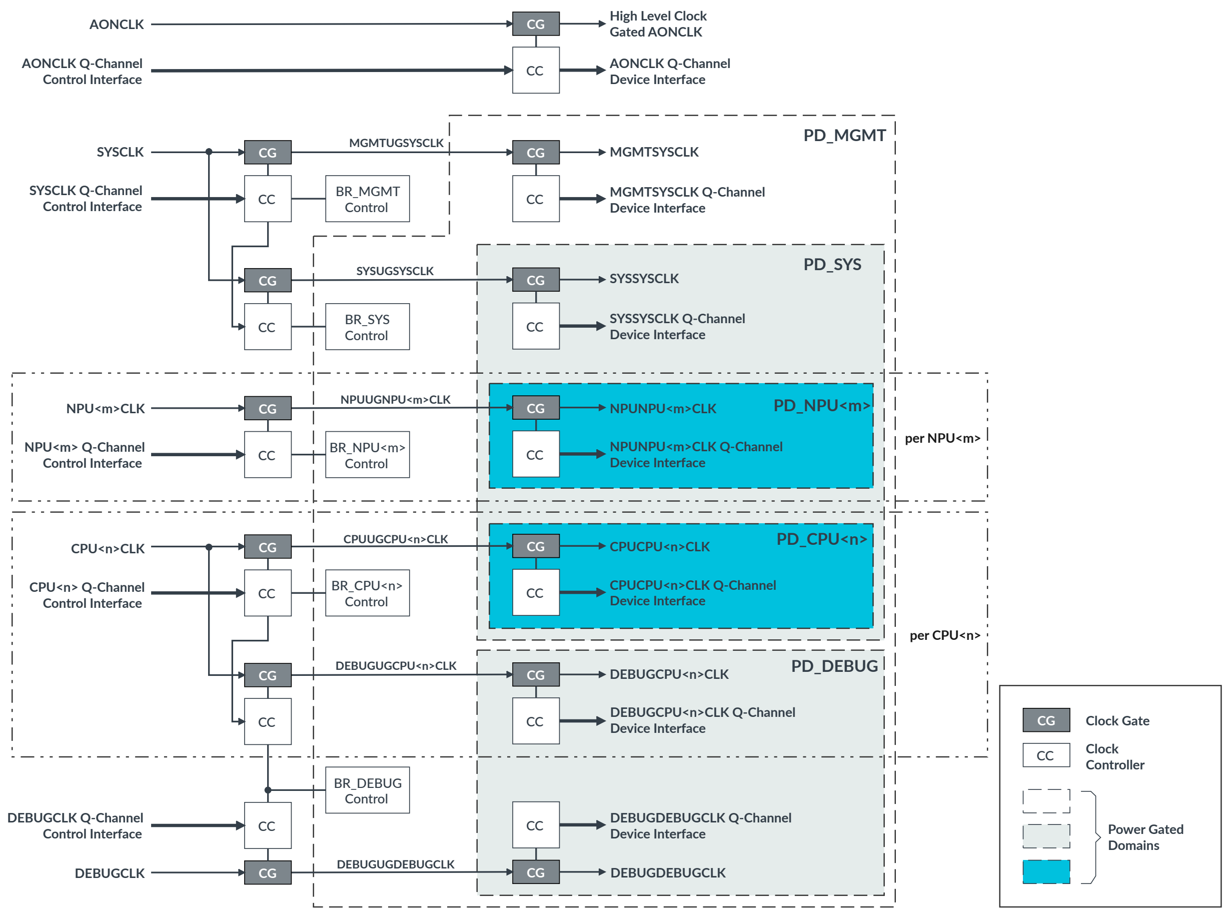
All blocks not in a Power Gated domain are in the always on PD_AON domain. For more information on power domains, see the Advanced level power infrastructure, Intermediate level power infrastructure. and Basic level power infrastructure topics.

In the example, clocks are often gated twice:

- In relation to the state of a power domain that the clock is driving.
- Internally within a domain depending on the activity within the domain.

An implementation of the CRSAS Ma2 can have varying levels of gating, and it is **IMPLEMENTATION DEFINED**.

Figure 5-1: Clock tree example when PILEVEL=2


AONCLK
AONCLK Q-Channel Control Interface
CG
CC
High Level Clock Gated AONCLK
AONCLK Q-Channel Device Interface
PD_MGMT
SYSCLK
SYSCLK Q-Channel Control Interface
MGMTUGSYSCLK
BR_MGMT Control
MGMTSYSCLK
MGMTSYSCLK Q-Channel Device Interface
PD_SYS
SYSUGSYSCLK
BR_SYS Control
SYSSYSCLK
SYSSYSCLK Q-Channel Device Interface
NPU<m>CLK
NPU<m> Q-Channel Control Interface
NPUUGNPU<m>CLK
BR_NPU<m> Control
NPUNPU<m>CLK
PD_NPU<m>
NPUNPU<m>CLK Q-Channel Device Interface
per NPU<m>
CPU<n>CLK
CPU<n> Q-Channel Control Interface
CPUUGCPU<n>CLK
BR_CPU<n> Control
CPUCPU<n>CLK
PD_CPU<n>
CPUCPU<n>CLK Q-Channel Device Interface
per CPU<n>
DEBUGCPU<n>CLK
DEBUGUGCPU<n>CLK
PD_DEBUG
DEBUGCPU<n>CLK Q-Channel Device Interface
BR_DEBUG Control
DEBUGCLK Q-Channel Control Interface
DEBUGDEBUGCLK Q-Channel Device Interface
DEBUGCLK
DEBUGUGDEBUGCLK
DEBUGDEBUGCLK
Clock Gate
Clock Controller
Power Gated Domains

The CRSAS Ma2 provides clock source force registers that request the clock sources to continue clock generation regardless of the state of the system. These clock source forces do not affect the power or high-level clock gating provided within the system. For more information, see CLOCK_FORCE.

The following figures show example clock structures that illustrate how the output clocks are related to the input clocks and their power domain relationships when PILEVEL=1 and PILEVEL=0. As the number of power domains are merged and are reduced, gating of each clock in relation to power domains are also merged as shown.

Figure 5-2: Clock tree example when PILEVEL=1

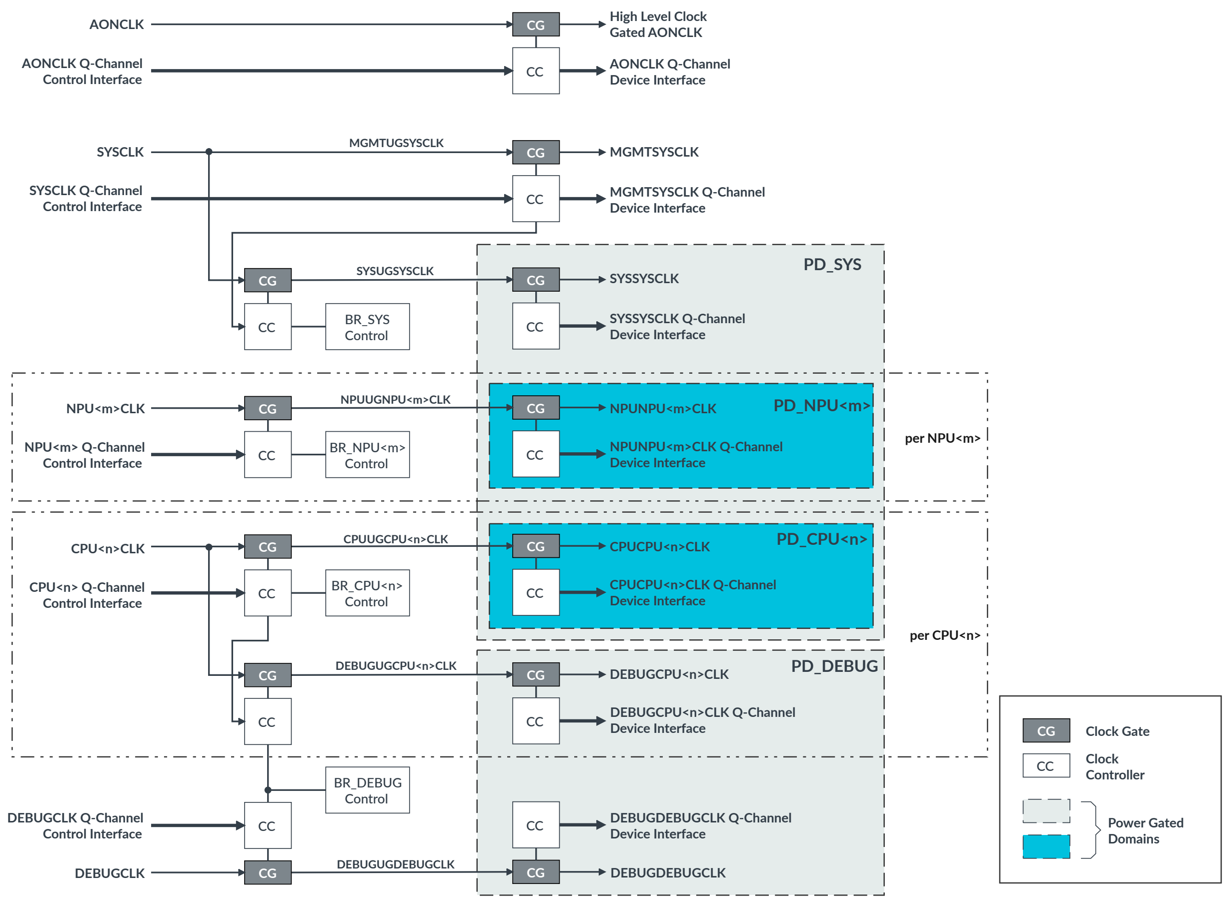
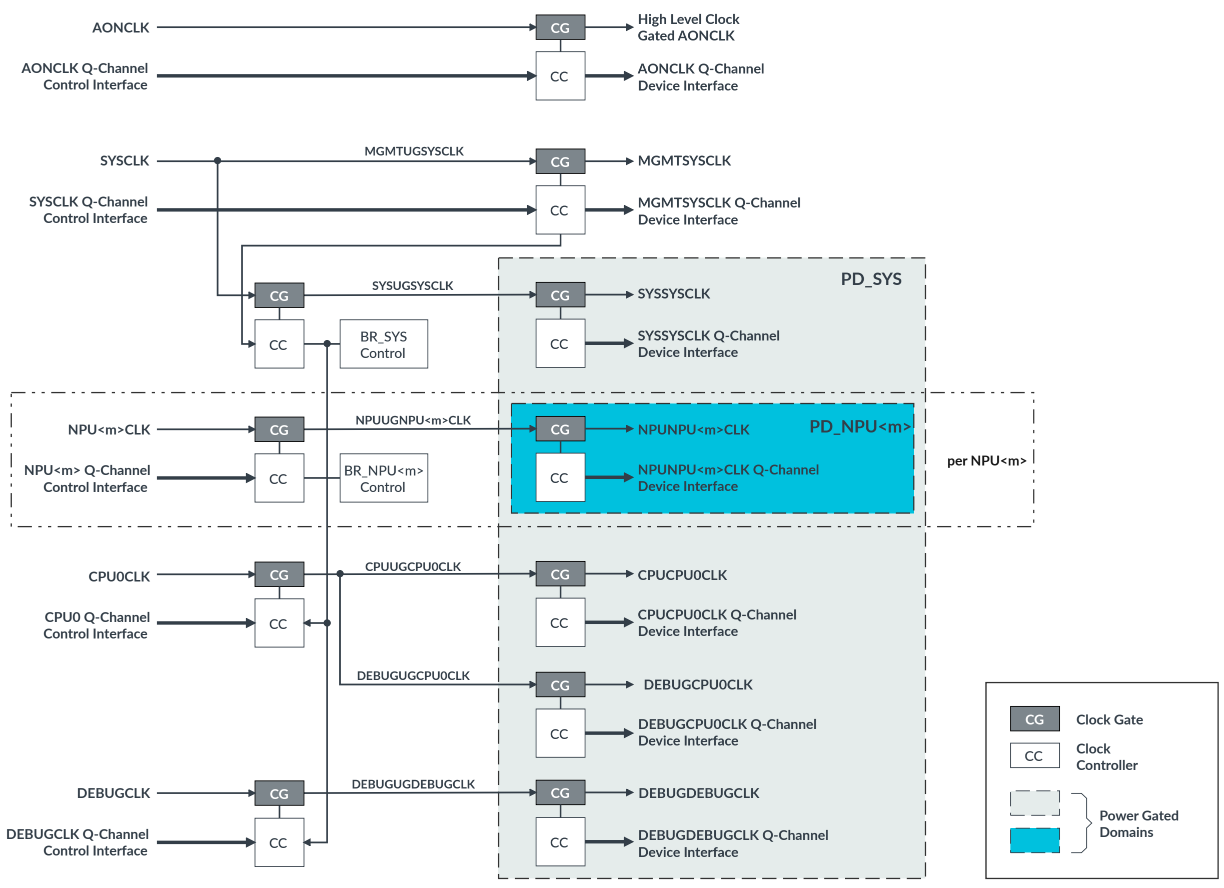

AONCLK
AONCLK Q-Channel Control Interface
High Level Clock Gated AONCLK
AONCLK Q-Channel Device Interface
SYSCLK
SYSCLK Q-Channel Control Interface
MGMTUGSYSCLK
MGMTSYSCLK
MGMTSYSCLK Q-Channel Device Interface
PD_SYS
SYSUGSYSCLK
SYSSYSCLK
SYSSYSCLK Q-Channel Device Interface
BR_SYS Control
NPU<m>CLK
NPU<m> Q-Channel Control Interface
NPUUGNPU<m>CLK
BR_NPU<m> Control
PD_NPU<m>
NPUNPU<m>CLK
NPUNPU<m>CLK Q-Channel Device Interface
per NPU<m>
CPU<n>CLK
CPU<n> Q-Channel Control Interface
CPUUGCPU<n>CLK
BR_CPU<n> Control
PD_CPU<n>
CPUCPU<n>CLK
CPUCPU<n>CLK Q-Channel Device Interface
per CPU<n>
DEBUGUGCPU<n>CLK
PD_DEBUG
DEBUGCPU<n>CLK
DEBUGCPU<n>CLK Q-Channel Device Interface
BR_DEBUG Control
DEBUGCLK Q-Channel Control Interface
DEBUGDEBUGCLK Q-Channel Device Interface
DEBUGCLK
DEBUGUGDEBUGCLK
DEBUGDEBUGCLK
CG
CC
CG Clock Gate
CC Clock Controller
Power Gated Domains

Figure 5-3: Clock tree example when PILEVEL=0


AONCLK
AONCLK Q-Channel Control Interface
CG
CC
High Level Clock Gated AONCLK
AONCLK Q-Channel Device Interface
SYSCLK
SYSCLK Q-Channel Control Interface
MGMTUGSYSCLK
MGMTSYSCLK
MGMTSYSCLK Q-Channel Device Interface
SYSUGSYSCLK
PD_SYS
SYSSYSCLK
BR_SYS Control
SYSSYSCLK Q-Channel Device Interface
NPU<m>CLK
NPU<m> Q-Channel Control Interface
NPUUGNPU<m>CLK
BR_NPU<m> Control
NPUNPU<m>CLK
PD_NPU<m>
NPUNPU<m>CLK Q-Channel Device Interface
per NPU<m>
CPU0CLK
CPU0 Q-Channel Control Interface
CPUUGCPU0CLK
CPUCPU0CLK
CPUCPU0CLK Q-Channel Device Interface
DEBUGUGCPU0CLK
DEBUGCPU0CLK
DEBUGCPU0CLK Q-Channel Device Interface
DEBUGCLK
DEBUGCLK Q-Channel Control Interface
DEBUGUGDEBUGCLK
DEBUGDEBUGCLK
DEBUGDEBUGCLK Q-Channel Device Interface
Clock Gate
Clock Controller
Power Gated Domains

5.2 Reset infrastructure

The CRSAS Ma2 reset infrastructure defines existing system level resets scopes, input and output signals.

The CRSAS Ma2 defines the following system level reset scopes:

- Power-on reset
- Cold reset
- Warm reset

It is not possible to have a Power-on reset without also having a Cold reset. It is not possible to have a Cold reset without also having a Warm reset.

On Power-on reset, registers that have a defined reset value contain that value. On Cold reset, it is the same as on Power-on, reset except the RESET_SYNDROME remains unchanged. On Warm reset, it is the same as on Cold reset, except debug related registers and parts of the power clock management infrastructure remain unchanged, as specified in this document. Local derivatives of these resets may exist per power or clock domain.

Reset distribution defines how the output resets are related to the input resets and reset requests, and which power domains the reset outputs primarily drive.

The CRSAS Ma2 has the following reset input signal to reset the subsystem:

nPORESET

This is the active-LOW Cold reset input for the system. We recommend that the reset is asserted for one or more SLOWCLK cycles.

The CRSAS Ma2 also has the following reset-related request and handshake signals:

nSRST

This input, typically from an external debugger, is an active-LOW system-wide reset request.

This reset request, when initially asserted, results in a Cold reset being applied to the subsystem and then subsequently removed. However, while the nSRST is asserted, the CPUWAIT inputs of all processors are held HIGH, preventing all processor cores from booting until the nSRST is de-asserted. This occurs regardless of the register setting in the CPUWAIT register. For more information on the nSRST, see *Arm® Debug Interface Architecture Specification ADIv6.0*.

While the minimum duration of the assertion of the nSRST is **IMPLEMENTATION DEFINED**, we strongly recommend that the nSRST input be at least three SLOWCLK cycles long, especially if the nSRST is asynchronous to the SLOWCLK. The reason is that the nSRST is likely to be treated as a reset request signal that needs to be resynchronised before use, rather than an actual reset signal. It can then be held LOW for as long as is required to hold off the processor from execution.

RESETREQ

This reset request input allows external expansion logic to request for a Cold reset.

After it is asserted, the signal must be held HIGH until the reset occurs on the nCOLDRESETAON, which clears this input. This reset is expected to be driven by expansion logic that is hosted by the subsystem.

HOSTRESETREQ

This reset request input allows a higher-level entity to reset the system.

A higher-level entity can be, for example, a host system that this subsystem is subservient to, which needs to have control over the reset of this system. Asserting this request causes a Cold reset. Holding this request HIGH maintains the subsystem in Cold reset.

Since this input, like the nSRST, is a request rather than a direct reset input, we strongly recommend that it is asserted for at least three the duration of SLOWCLK cycles.

WARMRESETREQ

When this Warm reset request input signal is set to HIGH, it requests the reset logic to perform a Warm reset of the system. Together, both the WARMRESETREQ and the WARMRESETACK operate a four phase handshake.

This handshake allows any external logic to request a system Warm reset and to determine when Warm reset occurs. If this signal is not used, you must tie the WARMRESETREQ to LOW. When the WARMRESETREQ is set to HIGH, the WARMRESETACK will also transition to HIGH, indicating that a Warm reset has occurred. To perform another Warm reset, you must clear the WARMRESETREQ and wait for the WARMRESETACK to transition to LOW before asserting the WARMRESETREQ again.

The existence of this signal and the associated acknowledge signal is **IMPLEMENTATION DEFINED**.

WARMRESETACK

This Warm reset acknowledges input signal works with the WARMRESETREQ to notify that a requested warm reset has occurred. Together, both the WARMRESETREQ and the WARMRESETACK operate a four phase handshake.

The existence of this signal and the associated request signal is **IMPLEMENTATION DEFINED**.

The following outputs are provided for expansion logic hosted by the subsystem:

nCOLDRESETAON

This PD_AON domain reset output is the Cold reset signal intended to be used by the subsystem expansion logic. This reset merges other reset sources that are required to cause Cold reset. See Power-on and Cold reset handling.

nWARMRESETAON

The subsystem generates this signal to perform a system Warm reset. The scope for this reset is a subset of the nCOLDRESETAON and is also asserted when the system is in WARM_RST state. It is expected to reset all non-debug related expansion logic that reside in the PD_AON power domain.

nCOLDRESETMGMT

The scope for this reset is a subset of the nCOLDRESETAON and is also asserted when PD_MGMT power domain is in lower power (OFF) states. When PILEVEL = 2, it is expected to be used in the PD_MGMT power domain. When PILEVEL < 2, it is expected to be used to reset logic that was in the PD_MGMT that is merged into PD_AON.

nWARMRESETMGMT

The scope for this reset is a subset of the nCOLDRESETMGMT and is also asserted when the system is in WARM_RST state. When PILEVEL = 2, this reset is used to reset all non-debug related expansion logic in the PD_MGMT power domain. When PILEVEL < 2, this reset is used to reset all non-debug related expansion logic in the PD_AON power domain that was merged from PD_MGMT.

nCOLDRESETSYS

The scope for this reset is a subset of the nCOLDRESETAON and is also asserted when PD_SYS power domain is in lower power (MEM_RET/OFF) states. It is expected to reset all expansion logic in the PD_SYS power domain.

nWARMRESETSYS

The scope for this reset is a subset of the nCOLDRESETSYS and is also asserted when system is in WARM_RST state. It is expected to reset all non-debug related expansion logic in the PD_SYS power domain.

nCOLDRESETDEBUG

The scope for this reset is a subset of the nCOLDRESETAON and is also asserted when PD_DEBUG power domain is in lower power (OFF) state. When PILEVEL > 0, it is expected to reset all expansion logic in the PD_DEBUG power domain. When PILEVEL = 0, it is expected to be used to reset all expansion logic that would have been in the PE_DEBUG power domain that is now merged into PD_SYS power domain.

nCOLDRESETCPU<n>

The scope for this reset is a subset of the nCOLDRESETAON and is also asserted when PD_CPU<n> power domain is in lower power (MEM_RET/OFF) states. It is expected to reset all expansion logic in the PD_CPU<n> power domain. However, when PILEVEL = 0, it is expected to be used to reset logic that would have been in the PD_CPU0 power domain that is merged into PD_SYS power domain.

nWARMRESETCPU<n>

The scope for this reset is a subset of the nCOLDRESETCPU<n> and is also asserted when system is in WARM_RST state. It is expected to reset all non-debug related expansion logic in the PD_CPU<n> power domain. However, when PILEVEL = 0, it is expected to be used to reset non-debug logic that would have been in the PD_CPU0 power domain that is merged into PD_SYS power domain.

nCOLDRESETDEBUGCPU<n>

The scope for this reset is a subset of the nCOLDRESETAON and is also asserted when PD_DEBUG system is in lower power (OFF) state. It is expected to reset all debug related expansion logic associated with debug logic of each processor core that resides in the PD_DEBUG power domain. However, when PILEVEL = 0, it is expected to be used to reset the CPU debug logic that would have been in the PD_DEBUG power domain that is merged into PD_SYS power domain.

The following signals are provided for internal use and defined for reference and clarity:

nWARMRESETNPU<m>

The scope for this reset is a subset of the nCOLDRESETAON and is also asserted when system is in WARM_RST state and when PD_NPU<m> is in lower power (OFF) state. It is expected to reset all logic in the PD_NPU<m> power domain. This reset must exist when NUMNPU > 0.

LCMRSTREQ

Lifecycle Manager (LCM) reset request. This reset request signal is connected to the sp_rst_req output signal of the LCM. When this signal is asserted by the LCM, the Warm reset combiner performs a system wide Warm reset. The Key Management Unit

(KMU), which normally is only affected by Cold reset, must also be Warm reset when the LCMRSTREQ is asserted. This signal is only asserted during Secure Provisioning (SP). Since the LCM and therefore the sp_rst_req output signal is not on Warm reset, it remains set while SP is active until the LCM is reset by the nCOLDRESETAON. Therefore, the Warm reset combiner is only sensitive to its positive edge. This signal must exist when LCM_KMU_SAM_PRESENT = 1. If LCM_KMU_SAM_PRESENT = 0, it is equivalent to the LCMRSTREQ being tied LOW.

SAMWRSTREQ

Security Alarm Manager (SAM) Warm reset request. This reset request signal is assigned by the Security Alarm Manager Response Action Settings. When this signal is asserted by the SAM, the Warm reset combiner performs a system wide Warm reset. Since the SAM and therefore the response action outputs are not on Warm reset mode, actions can remain set during and post Warm reset. Therefore, the Warm reset combiner is only sensitive to its positive edge. This signal must exist when LCM_KMU_SAM_PRESENT = 1. If LCM_KMU_SAM_PRESENT = 0, it is equivalent to the SAMWRSTREQ being tied LOW.

SAMCRSTREQ

SAM Cold reset request. This reset request signal is assigned by the Security Alarm Manager Response Action Settings. When this signal is asserted by the SAM a system wide Cold reset is performed. Since the parts of the SAM and therefore the response action outputs are not on Cold reset, actions can remain set during Cold reset. Therefore, the system is only sensitive to its positive edge. This signal must exist when LCM_KMU_SAM_PRESENT = 1. If LCM_KMU_SAM_PRESENT = 0, it is equivalent to the SAMCRSTREQ being tied LOW.

The following status outputs are provided to indicate the status of four events that can be used to generate a Cold reset of the system.

NSWDRSTREQSTATUS

Non-secure watchdog reset request status. This output is '1' when the Non-secure watchdog is raising a reset request and RESET_MASK.NSWDRSTREQEN is '1'. When set to HIGH, it does not return to LOW, unless a reset occurs clearing this status.

SWDRSTREQSTATUS

Secure watchdog reset request status. This output is '1' when the Secure watchdog is raising a reset request. When set to HIGH it does not return to LOW unless a reset occurs on the nCOLDRESETAON, clearing this status.

RESETREQSTATUS

Hardware reset request status. This output is set to '1' when the RESETREQ input is '1'. When set to HIGH, it is not cleared unless the system is reset on the nCOLDRESETAON, which restores the register field to '0'.

SAMCRSTREQSTATUS

SAM Cold reset request status. This reset request signal is assigned by the Security Alarm Manager Response Action Settings. When this signal is asserted by the SAM, a system wide Cold reset is performed. Since the parts of the SAM and therefore the response action outputs are not on Cold reset, actions can remain set during Cold reset. Therefore, if this signal is used to generate the Cold reset externally, the generation logic must be sensitive only to its positive edge. This signal must exist when LCM_KMU_SAM_PRESENT

= 1 and COLDRESET_MODE = 1. If LCM_KMU_SAM_PRESENT = 0, it is equivalent to SAMCRSTREQ being tied LOW.

SWCOLDRSTREQSTATUS

Software reset request status. This output is '1' when SWRESET.SWCOLDRESETREQ is set to '1'. When set to HIGH, it must not be cleared unless the system is reset on nCOLDRESETAON, which restores the register field to '0'.

SSWDRSTREQSTATUS

Secure privileged SLOWCLK watchdog reset request status. This output is '1' when the Secure privileged SLOWCLK watchdog is raising a reset request. When set to HIGH, it does not return to LOW unless a reset occurs on the nCOLDRESETAON, clearing this status.

Figure 5-4: Reset structure example when PILEVEL = 2 on page 51 shows an example reset structure when PILEVEL = 2. The figure shows how the output resets are related to the input resets and reset request, and which power domain the reset output primarily drives. The diagram does not show infrastructure for reset resynchronization and how the resets are used within each domain. It shows the nPORESETAON reset signal. The reset signal is not an output of the subsystem and is only used within the system to reset the RESET_SYNDROME register and some part of the SAM and the corresponding Sensor Alarm interface.

For more information on the SAM resets, see *Arm® Security Alarm Manager (SAM) Specification*.

An implementation of the CRSAS Ma2 can have a different reset tree implementation but the relationships as specified for the reset signals must remain the same.

Figure 5-4: Reset structure example when PILEVEL = 2

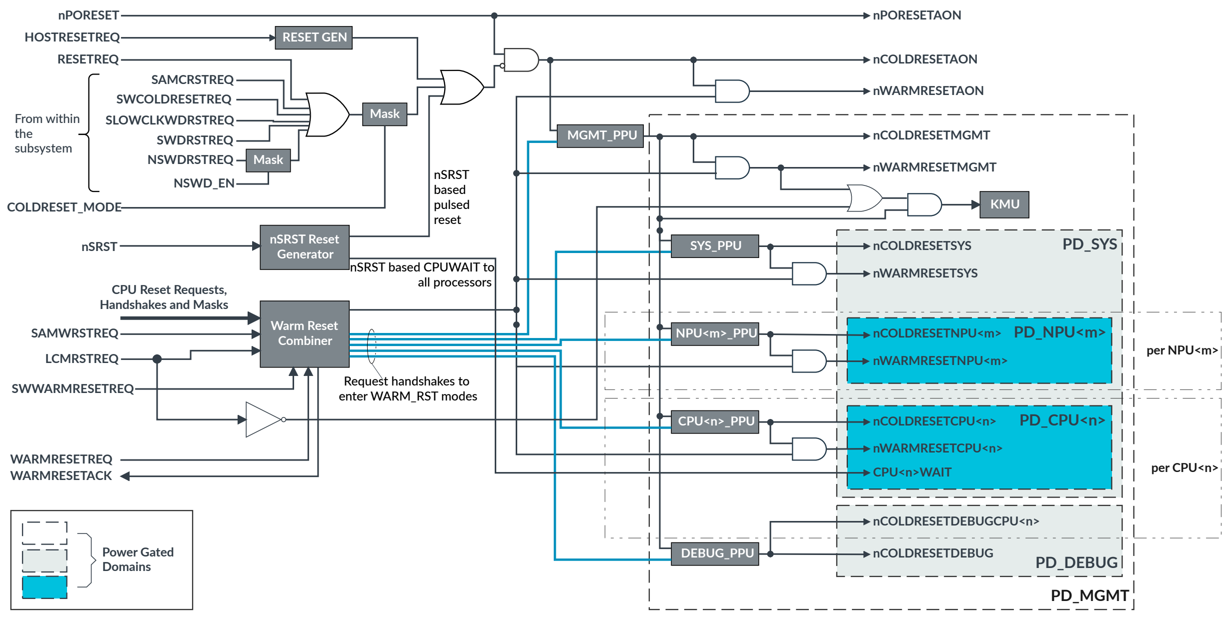


Figure 5-5: Reset structure example when PILEVEL = 1 on page 52 and Figure 5-6: Reset structure example when PILEVEL = 0 on page 52 show example reset structures when PILEVEL

= 1 and PILEVEL = 0. These show how reset generation is combined as the number of power domains are reduced.

Figure 5-5: Reset structure example when PILEVEL = 1

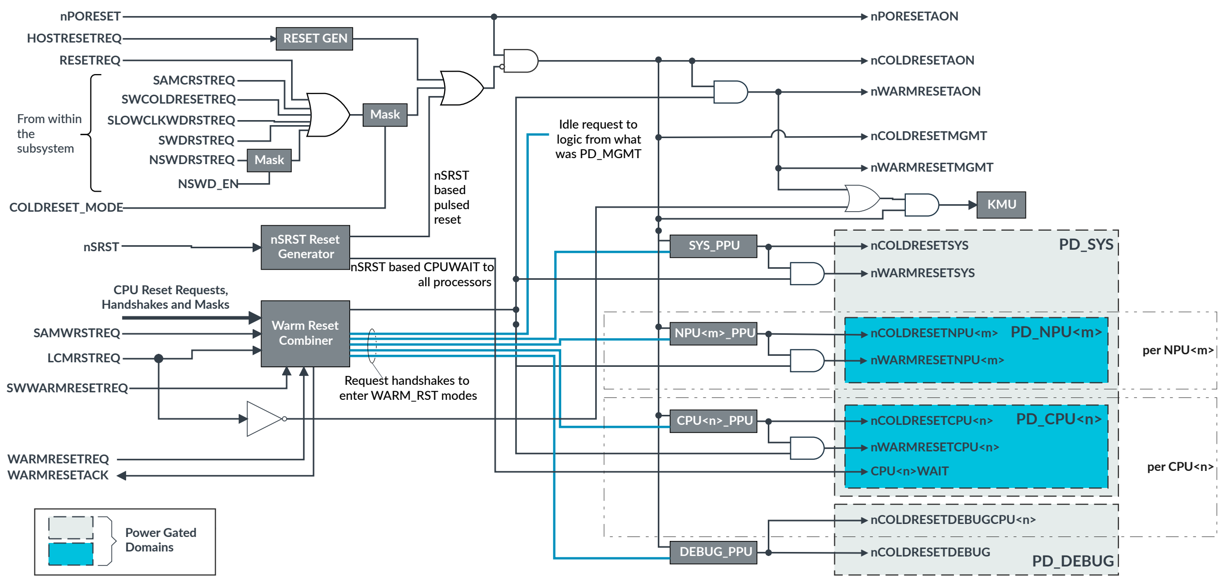
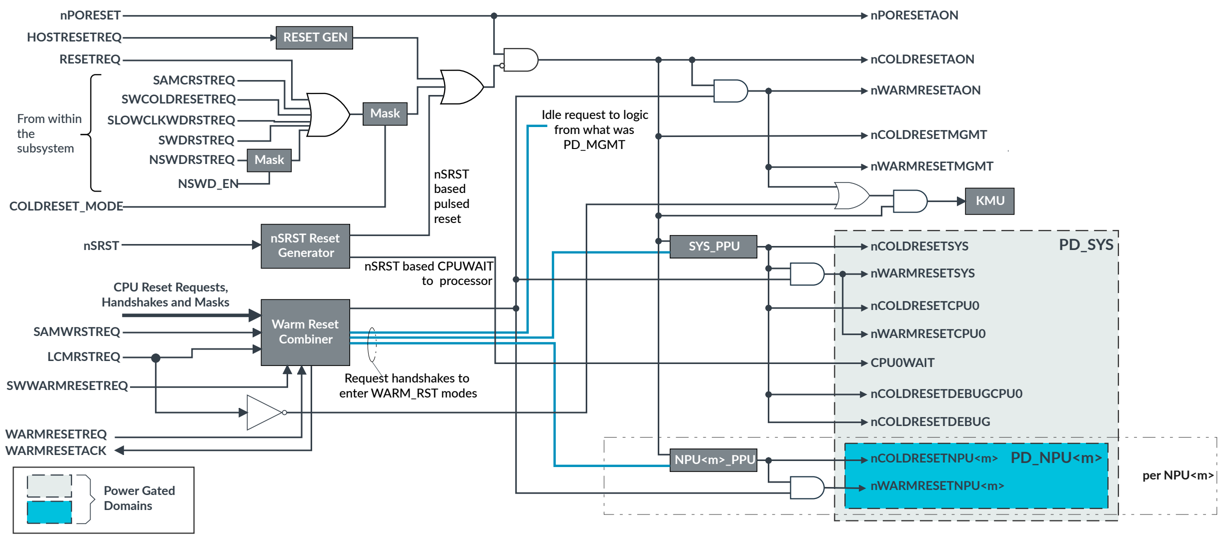


Figure 5-6: Reset structure example when PILEVEL = 0



5.2.1 Power-on and Cold reset handling

The input nPORESET is the power-on reset input that resets all registers in the design. This includes resetting the reset syndrome register.

For more information on the register, see Reset Syndrome.

The nPORESET is combined with the masked and combined reset requests to generate the internal combined Cold reset, which is available for use by expansion through the nCOLDRESETAON output. The requests are from the following:

- If COLDRESET_MODE = 0, then:
    - Reset requests from every watchdog timer, through a relevant mask if it exists for each.
    - Reset request input RESETREQ.
    - Reset request through the SWRESET.SWCOLDRESETREQ register value.
- Reset request input HOSTRESETREQ.
- Reset generated from nSRST request by using negative-edge detection first and then stretching the result.

The nCOLDRESETAON signal resets almost all logic within the system except for the RESET_SYNDROME registers.

Note

In each power domain that is directly controlled by a PPU that resides in the PD_AON or PD_MGMT domain, the PPU is reset using the nCOLDRESETAON or the nCOLDRESETMGMT respectively. Each PPU then generates the Cold reset that is used within the power domain it controls. For example, the PPU for PD_SYS is responsible for generating the nCOLDRESETSYS.

When COLDRESET_MODE = 1, it allows the system to ignore the internal Cold reset requests like watchdog timers, the RESETREQ, the SAMCRSTREQSTATUS and the SWRESET.SWCOLDRESETREQ based reset when generating Cold reset. This essential occurs if the Cold reset control is owned by a different system.

When COLDRESET_MODE = 1, if the value of various statuses is ‘1’, the external entity must cause a reset by using the HOSTRESETREQ input in a short time. The duration of time during which the external entity must cause the reset is **IMPLEMENTATION DEFINED**. The list statuses that can cause this are as follows:

- SAMCRSTREQSTATUS
- NSWDRSTREQSTATUS
- SWDRSTREQSTATUS
- SSWDRSTREQSTATUS
- RESETREQSTATUS
- SWCOLDRSTREQSTATUS

Caution

When COLDRESET_MODE = 1 and reset is requested by one of the reset status outputs, failing to reset the system with the HOSTRESETREQ input can cause system deadlock or even compromise security.

5.2.2 CPU reset handling

The nPORESET reset input of each CPU<n> is driven by the PPU that controls each processor core power domain, PD_CPU<n>. This PPU itself is reset using the nCOLDRESETMGMT.

A Warm reset mode is driven from Warm reset generation logic that resides in the PD_AON domain. This, along with all the PPUs in the system, is used to force the system to idle before driving Warm reset that includes the nSYSRESET input of the CPU<n>. The nSYSRESET signal of each processor is generated when the system enters WARM_RST state. This ensures that the logic in PD_CPU<n> power domain is quiescent when that occurs.

If the reset is the result of a Cold reset request from the nSRST input. After a momentary Cold reset, CPUWAIT input of all processors is forced HIGH as long as the nSRST is held LOW. This stops the processor from starting execution until the nSRST is released. This in turn allows a debugger to hold the processor core from execution after reset, while it uses the DAP to perform debug operations.

5.2.2.1 Boot after reset

After resets, including power-on reset, all processors boot using the boot address values defined in the Initial Secure Reset Vector registers (INITSVTOR<n>) in the system control registers.

For more information about the register, see INITSVTOR<n>.

The default address is **IMPLEMENTATION DEFINED**, but we recommend that the default is set to point to:

- The boot ROM, if it exists at `0x1100_0000`. This is in keeping with the example in system boot flow.
- If the ROM does not exist, it should instead point to an immutable, Secure, non-volatile storage that contains the First Stage Boot Loader (FSBL). For example, it could point to `0x1200_0000`, which is mapped to a Secure Flash memory storing boot code by the Manager Code Main Expansion interface.

These addresses can be modified by the software before subsequent Warm reboots of the processors.

The TrustZone for Armv8-M states that boot must start from a Secure memory space. At boot, all VM is Secure only. The software must change or restore the settings in the MPC to release memory for Non-secure world use.

The CPUWAIT input to each core can force each processor to wait before executing the instruction. Each processor in the system has an associated CPU<n>WAIT register that controls if it starts running its boot code when it wakes. There is also a CPU<n>WAITCLR input for each CPU<n> to allow an external entity to clear the associated CPU<n>WAIT register bit.

For more information, see CPUWAIT.

5.2.3 NPU reset handling

If NUMNPU > 0, each nRESET input of the NPUs is driven by the PPU that controls the NPU power domain PD_NPU<m>. In this case Warm reset is driven from Warm Reset Generation logic that resides in the PD_AON domain. This, along with all the PPUs in the system, is used to force the system to idle before driving Warm reset.

The PPU itself is reset using the nCOLDRESETMGMT.

5.2.3.1 NPU security and privilege from reset

When the NPU reset is released by the PPU controlling the NPU power domain PD_NPU<m>, the NPU<m> samples the signals driven by NPUSPPORSL.SP_NPU<m>PORSL to determine its default security level. It also samples the signals driven by NPUSPPORPL.SP_NPU<m>PORPL or NPUNSPORPL.NS_NPU<m>PORPL to determine its default privilege level.

For more details, see NPUSPPORPL and NPUNSPORPL.

For information on how the NPU is expected to transition between security and privilege levels, see NPU security mapping.

5.2.4 Warm reset generation

The system provides a Warm reset for each power domain, which is available for the expansion system on the nWARMRESETAON signal.

Warm reset is generated by first merging system reset requests from various locations, if they exist, after masking each with the values in the RESET_MASK register. The request locations are as follows:

- All the CPU<n> in the system
- SWCOLDRESETREQ field of the SWRESET register
- SAM
- LCM

The merged reset request is then used to request all the PPUs in the system to enter Warm reset state for logic that is not under the control of a PPU. For example, in the PD_AON, they also need to be requested to enter an idle state. When all the PPUs have entered Warm reset state, the WARM_RST system power state, all logic that are to enter Warm reset mode or can be affected indirectly by Warm reset mode are idle. Then the nWARMRESETAON is asserted. Each lower hierarchical domain has its local version of Warm reset signal to perform Warm reset.

Note

The PPUs are not reset by Warm reset.

The nCOLDRESETAON signal also drives the nWARMRESETAON, but this request is immediate and without requesting for all the PPUs to enter Warm reset state. Asserting the nCOLDRESETAON also resets all the PPUs in the system.

At implementation, the MGMT_PPU can act as the controller to bring all logic that are in the PD_MGMT to enter idle state. It can also request for all other PPUs to enter Warm reset state. When PILEVEL < 2, the reset generated by the MGMT_PPU can also instead be used as Warm reset mode signal to all logic. This means that the MGMT_PPU can exist when PILVEL < 2 but is only used for the sequencing and generation of Warm reset mode. MGMT_PPU is only ever in ON or WARM_RST state. the MGMT_PPU effectively implements most of Warm reset combiner logic.

When PILEVEL = 2, the PD_MGMT can be used in the same way. Except, Warm reset mode generated by the MGMT_PPU must never be used to Warm reset mode logic in the PD_AON because the PD_MGMT now supports more power states. When cycling through OFF or RETENTION state, the PD_MGMT logic can be reset, and this reset must not be applied to PD_AON.

5.3 Read-only memory

The CRSAS Ma2 supports an optional single bank of Read-Only Memory (ROM) for boot code. If the the ROMADDRWIDTH configuration point is set to '0', the ROM is not present in the system. If present, the memory size is configurable and must obey the following:

- A 16MB region of memory at `0x1100_0000` to `0x11FF_FFFF` is assigned to the boot ROM only if the ROM exist.
- The size of the ROM is powers of 2.
- The size is defined using the ROMADDRWIDTH configuration point and is equal to $2^{\text{ROMADDRWIDTH}}$ bytes.
- To reduce the risk of ROM code issues, it is highly recommended to minimise the content of the ROM and put other functionality in ROM loaded code.
- The ROM forms a contiguous area of memory and is mapped to a starting address of `0x1100_0000`, for Secure access only.
- Access to any unused area of the 16MB region `0x1100_0000` to `0x11FF_FFFF` returns a bus error.

The ROM has a Main Interconnect Peripheral Protection Controller (MIPPC0) associated with it that provides protection on subordinate side and restricts the ROM to Secure privileged world. The PPC protected components can raise interrupt on security violation and respond with bus error or **RAZ/WI** depending on the global SECRESPCFG configuration.

For more information on PPC, see the *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual* and the *Arm® CoreLink™ SIE-300 AXI5 System IP for Embedded Technical Reference Manual*.

The CRSAS Ma2 supports no separate power domain for ROM bank, the Peripheral Protection Controller (PPC), and the SRAM Memory Controller (SMC). They reside in the PD_SYS power domain. The ROM follows the power states of the PD_SYS and runs on the SYSSYSCLK. For more information on power control of the PD_SYS, see BR_SYS power modes.

When the boot ROM does not exist, the memory region `0x1100_0000` to `0x11FF_FFFF` is assigned to the Manager Code Main Expansion interface.

5.4 Timers and watchdogs

The CRSAS Ma2 supports two main classes of timers and watchdogs: timestamp based timers and SLOWCLK AON based timers.

5.4.1 Timestamp-based timers

Timestamp-based timers use the timestamp value provided on the System Timestamp interface.

For more information on the interface, see System Timestamp interface.

The timestamp-based timers provided in the CRSAS Ma2 have the following features:

- Memory-mapped system timer with register access through the peripheral interconnect
- 64-bit timestamp input, generating events through comparison with a timer value
- An ‘auto-increment’ feature to support regular event generation
- Down counter emulation
- Maskable level interrupt generation

The timestamp-based watchdog timers are simplified timers that support the following:

- Memory-mapped system timer with register access through the peripheral interconnect
- 64-bit timestamp input, generating events through comparison with a timer value
- An ‘auto-increment’ feature to support refreshing the watchdog
- Watchdog reset request generation on double watchdog timeout
- Separate refresh register access frame to support refreshing from a different security or privilege level

See Timestamp-based timers registers and Timestamp-based watchdogs registers for details on the timer registers.

Four timestamp-based timers are provided along with two timestamp-based watchdog timers. These are mapped to the following addresses:

- Timer 0 at address `0x4800_0000` and aliased to `0x5800_0000`
- Timer 1 at address `0x4800_1000` and aliased to `0x5800_1000`
- Timer 2 at address `0x4800_2000` and aliased to `0x5800_2000`
- Timer 3 at address `0x4800_3000` and aliased to `0x5800_3000`

- Secure watchdog timer control frame at address `0x5804_0000`, and refresh frame at `0x5804_1000`
- Non-secure watchdog timer control frame at address `0x4804_0000`, and refresh frame at `0x4804_1000`.

Timer 0, Timer 1, Timer 2, and Timer 3 can be configured by the software to be a Secure or a Non-secure timer through the PPC, and the software can control that PPC through the Secure Access Configuration Register block.

Security mapping of both watchdog timers are permanently assigned: one as Non-secure and another as Secure. The control frames are permanently restricted to only privileged accesses while the refresh frames are configurable through PPC to be:

- privileged accessible only
- privileged and unprivileged accessible

All timers and watchdog timers generate interrupts, and the Non-secure watchdog can generate an additional interrupt on a second timeout event for notifying the Secure world. This allows the Secure world to handle the Non-Secure watchdog double timeout instead of directly applying reset to the system immediately. The Secure watchdog can request a reset of the system if double timeout occurs. The Non-secure watchdog can be configured by the software to do the same if necessary. By default this is not allowed.

All timestamp-based timers and watchdogs, except for Timer 3, reside in the PD_SYS power domain and are reset by the nWARMRESETSYS. Timer 3 resides in the PD_AON power domain and is reset by the nWARMRESETAON. Therefore, Timer 3 can generate interrupts to wake the system even if the system is in the Hibernation state where the PD_SYS is turned off or in retention.

5.4.2 SLOWCLK AON timers

The second class of timers and watchdogs are simple CMSDK based 32-bit timers that run on the SLOWCLK. They reside in the PD_AON power domain and are reset by the nWARMRESETAON. A single timer and a single Secure privileged watchdog are provided and are expected to be used when the system is in HIBERNATION{0-1}. In HIBERNATION{0-1} potentially only the SLOWCLK is available and running, and all other clocks are off.

These timers are mapped to the following addresses:

- SLOWCLK timer at address `0x4802_F000` and aliased to `0x5802_F000`.
- Secure privileged SLOWCLK watchdog timer at address `0x5802_E000`.

The SLOWCLK timer can be configured by the software to be Secure or Non-secure access only, and to have privileged or unprivileged access through the PPC that is controlled through registers in the Secure Access Configuration Register block and the Non-secure Access Configuration register block. The watchdog is Secure privilege access only. All CMSDK timers and watchdog timer generate interrupts, but the Secure watchdog can request a Cold reset of the system if double timeout occurs.

For more information on CMSDK timers and watchdog, see *Arm® Cortex®-M System Design Kit Technical Reference Manual*.

5.5 Message Handling Unit

When NUMCPU > 0, the CRSAS Ma2 implements two Message Handling Units (MHUs) to allow the processors to interrupt each other to pass message. Two MHUs are provided so that it is possible for the software to place one MHU in the Secure world and another in the Non-secure world. Both MHUs reside in the PD_SYS power domain and are reset using the nWARMRESETSYS.

See Message Handling Unit register map for details on the MHU registers. The MHUs in the system are expected to conform to the MHUv1 specification.

When NUMCPU = 0, there are no MHUs in the system.

5.6 Power Policy Units

The CRSAS Ma2 leverages the Power Policy Units (PPUs) for power control for each bounded region in the system. Each bounded region is a collection of power domains where their power states are controlled collectively, usually by a PPU.

For more information on PPU, see *Arm® Power Policy Unit Architecture Specification* and *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual*.

The following PPU Associations and Mandated Configurations table lists the mandatory configurations of all the PPUs in the system, which bounded region each controls, and what power domain each resides in. Other PPU configurations that are not listed here are **IMPLEMENTATION DEFINED**.

Note

While viewing the PPU Associations and Mandated Configurations table, note the following:

- If PILEVEL = 0:
  - SYS_PPU does not exist
  - CPU0_PPU is renamed as SYS_PPU.
- The NPU<m>_PPU only exists when NUMNPU > 0

Table 5-1: PPU associations and mandated configurations

PPU configuration	MGMT_PPU	DEBUG_PPU	SYS_PPU	CPU<n>_PPU	NPU<m>_PPU
Bounded Region Controlled by PPU	BR_MGMT	BR_DEBUG	BR_SYS	BR_CPU<n>	BR_NPU<m>
Power domain the PPU resides in	PD_AON	PD_MGMT If PILEVEL < 2, the PD_MGMT is merged into the PD_AON			
Reset signal used by the PPU	nCOLDRESETAON	nCOLDRESETMGMT			

PPU configuration	MGMT_PPU	DEBUG_PPU	SYS_PPU	CPU<n>_PPU	NPU<m>_PPU
Device interface type	P-Channel	P-Channel			
Default power policy (DEF_PWR_POLICY)	OFF: PILEVEL = 2 ON: PILEVEL != 2	OFF			
Default power mode dynamic transition enable (DEF_PWR_DYN_EN)	1	1			
Dynamic support	ON WARM_RST OFF	ON WARM_RST OFF	ON FULL_RET MEM_RST WARM_RST OFF	ON FULL_RET MEM_RST MEM_OFF FUNC_RST LOGIC_RST WARM_RST OFF	ON WARM_RST OFF
Default Operating Policy (DEF_OP_POLICY)	0	0	0	0	0
Number of Operating Modes (NUM_OPMODE_CFG)	0	0	$2^{NUMVMBANK}-1$	3	0
Operating Mode Active configuration (OP_ACTIVE_CFG)	0	0	1 if NUMVMBANK > 0, else 0	1	0
Default Operating Mode Dynamic Transition Enable (DEF_OP_DYN_EN)	0	0	1 if NUMVMBANK > 0, else 0	1	0
Default Perform Device Interface Handshake When Transition from ON to WARM_RST mode Enable (WARM_RST_DEVREQEN_CFG)	1	1	1	1	1

The PPUs are mapped to the Secure address space as defined in System Control Peripheral Region. The write accessibility of the PPU registers is controlled through the PWRCTRL.PPU_ACCESS_FILTER. When it is set to '1', the system blocks all write accesses to the PPUs by ignoring the writes. Except for the following registers, if they exist for each PPU:

- Interrupt Mask register, at address offset `0x030`
- Additional Interrupt Mask register, at address offset `0x034`
- Interrupt Status register, at address `0x038`
- Additional Interrupt Status register, at address `0x03C`

Access to the PPU is normally not required for normal operation because the CRSAS Ma2 is architected to use the PPU primarily in dynamic mode, where the request to enter or leave a power state is managed and handshaked using the Device interface of the PPU. Hence the only time access to PPUs might be required is for debug purposes.

When PWRCTRL.PPU_ACCESS_FILTER is set to '0', all the PPU registers are freely accessible to the Secure world.

5.7 Peripheral Protection Controllers

The Peripheral Protection Controllers (PPCs) in the system enable the software to control whether a peripheral is accessible to the Secure or Non-secure world. It also enables the software to control privileged access or unprivileged access.

The PPC protected peripherals, depending on the global SECRESPCFG configuration, can: - Raise interrupt on security violation - Raise response with bus error or RAZ/WI

For more information, see *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual* and *Arm® CoreLink™ SIE-300 AXI5 System IP for Embedded Technical Reference Manual*.

In the CRSAS Ma2, peripherals that are aliased to two memory areas, one Secure and another Non-secure, are protected by PPCs. The PPC defines which region the peripheral resides in.

Within the CRSAS Ma2, subordinate peripheral interfaces are protected using PPCs. These are controlled by the Secure Access Configuration register block and Non-secure Access Configuration register block. These registers also control Security Control Expansion signals to drive external PPCs.

The CRSAS Ma2 defines two groups of peripherals protected behind Peripheral Protection Controllers. These groups are as follows:

Peripheral Interconnect Peripheral Protection Controller 0 (PIPPC0)

This group includes the following peripherals:

- Memory Protection Controller configuration register block
- Secure Access Configuration register block
- Non-secure Access Configuration register block
- Timestamp-based Watchdog Control Frames
- Timestamp-based Watchdog Refresh Frames
- All timestamp based timers
- Message Handling Units (MHUs)
- Debug System Access
- Watchdog Refresh Frames. Secure or Non-secure mapping of Watchdog Refresh Frames is fixed, only their privilege levels are configurable.
- All NPUs when NUMNPU > 0
- See also the PERIPHNSPPC0, PERIPHSPPPC0, PERIPHNSPPPC0, NPUNSPORPL, NPUSPPORSL, and NPUSPPORPL registers.

Peripheral Interconnect Peripheral Protection Controller 1 (PIPPC1)

This group includes the following peripherals:

- System Control register block
- SLOWCLK watchdog timer
- PPUs
- SLOWCLK timers
- Lifecycle Manager when LCM_KMU_SAM_PRESENT = 1
- Security Alarm Manager when LCM_KMU_SAM_PRESENT = 1
- Key Management Unit when LCM_KMU_SAM_PRESENT = 1
- See also the PERIPHNSPPC1, PERIPHSPPPC1, and PERIPHNSPPPC1 registers.

Caution

Exercise caution when permitting unprivileged access to bus manager in the system, for example, DMA or NPUs. If unprivileged access is permitted, then unprivileged code has the potential to use these managers to access and modify privileged memory or peripherals. We recommend that only privileged programming access is permitted to these managers.

For more information, see Secure Access Configuration Register block.

5.8 Memory Protection Controllers

Memory Protection Controllers (MPCs) in the system partition memory into pages and allow the software to define if each region is Secure or Non-secure.

For more information, see *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual* and *Arm® CoreLink™ SIE-300 AXI5 System IP for Embedded Technical Reference Manual*.

In the CRSAS Ma2, each memory page that is protected by the MPC is aliased to two memory areas, one Secure, and another Non-secure. Depending on the security attribute defined for that page in the MPC by the software, the page either only exists in the Secure region or the Non-secure region.

MPC protected memory pages can raise interrupt on security violation and response with bus error or **RAZ/WI** depending on the MPC or the SECRESPCFG register settings. The combined interrupt status of internal and external MPCs is available in the SECMPCINTSTAT register.

A single MPC is provided for each VM bank, and each is mapped into the main memory as defined in the Peripheral region.

5.9 Manager Security Controllers

The Manager Security Controllers (MSCs) in the system transform memory transactions issued by non-processor managers that do not support Arm TrustZone for Armv8-M. The goal of the transformation is to make the memory transactions suitable to Arm TrustZone for Armv8-M systems.

Within the CRSAS Ma2, manager peripheral interfaces are protected using MSCs. These are controlled by the Secure Access Configuration Register block. MSC protected managers can raise interrupt on security violation and response with bus error or **RAZ/WI** depending on the global SECRESPCFG configuration.

For more information, see *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual* and *Arm® CoreLink™ SIE-300 AXI5 System IP for Embedded Technical Reference Manual*.

5.10 Lifecycle Manager

The CRSAS Ma2 defines an LCM module that manages a set of RoT keys, debug controls, and security states stored in an OTP array. The LCM controls the life cycle states of the system, debug control signals, and RoT keys.

The LCM also interacts with the Key Management Unit (KMU). The LCM passes the RoT keys from the Non-Volatile One Time Programmable Memory into the KMU once per Cold reset. The LCM supports the Secure provisioning of confidential assets into the OTP in an untrusted environment. For more information, see Secure Asset Provisioning Flow.

For more information on the LCM and the KMU, see *Arm® Key Management Unit (KMU) Specification* and *Arm® Lifecycle Manager (LCM) Specification*. The configuration options of the LCM are defined in Lifecycle Manager related configurations and *Arm® Lifecycle Manager (LCM) Specification*.

The LCM and its interfaces only exist when LCM_KMU_SAM_PRESENT = 1.

The LCM interfaces are integrated in the CRSAS Ma2 as follows:

- APB3 subordinate interface to program the LCM registers.
    - This interface is PPC protected, always Secure access only, and the privileged/unprivileged access is configurable, see PERIPHSPPPC1.
    - Sparse writes are not allowed, and must result in error response and memory intact.
    - The programming base address is defined in the Peripheral region.
    - This interface also provides access to parts of the NVM (OTP).
- The Lifecycle Manager Non-Volatile Memory interface is exported to the top level.
- The direct key APB3 manager interface is directly connected to the KMU for exporting keys.
- Lifecycle and debug control signals are exported to the top level. See Lifecycle Manager Lifecycle Indication interface and Lifecycle Manager Debug Control Unit interface.
- Lifecycle Manager LFSR seed interface for loading a low-quality entropy random number into LCM is exported to the top level.
- Error indication connects to the SAM and exported to the top level. See Lifecycle Manager Lifecycle Indication interface.
- The LCM clock is synchronous to the MGMTSYSCLK.

- The LCM resides in the PD_MGMT power domain.
- The LCM reset is the nCOLDRESETMGMT.

The context of the LCM must be saved and retained (or restored) over HIBERNATION0. All signals that are driving the PD_AON logic have to be latched and hold during HIBERNATION0.

5.10.1 LCM Debug Control Unit

The LCM Debug Control Unit provides Lifecycle dependent control to the Debug Authentication interface and the Lifecycle Manager Debug Control Unit interface outputs. The Debug Authentication interface also has a separate override mechanism to force disable debug features during boot. This is controlled through the LCM_DCU_FORCE_DISABLE register, using the DCUEN[31:0] signals.

For more information on the outputs, see Debug Authentication interface and Lifecycle Manager Debug Control Unit interface.

For more informaton on the register, see LCM_DCU_FORCE_DISABLE.

Note

The LCM_DCU_FORCE_DISABLE[21:0] are used in pairs, where each pair of bit values represent a single debug control bit. Similarly, the DCUEN[21:0] signals are also used in pairs, where each pair of bit values represent a single debug control bit.

The LCM_DCU_FORCE_DISABLE[31:22] bits are used individually, where each bit represents a single debug control bit. Similarly, the DCUEN[31:22] signals are used individually, where each bit represents a single debug control bit.

All values of the LCM_DCU_FORCE_DISABLE_INIT configuration option must be set to force the disabling of all used Debug Authentication interface and the DCUEN[31:22] signals.

Assignment of DCUEN controls

The LCM DCUEN[31:0] signals are reserved for internal use and are not exposed to the expansion through the Lifecycle Manager Debug Control Unit interface.

The number of reserved signals is aligned to the number of signals that might be disabled by the LCM_DCU_FORCE_DISABLE register.

DCUEN[21:0]

The LCM DCUEN[21:0] signals and the LCM_DCU_FORCE_DISABLE[21:0] control bits are reserved to control the Debug Authentication interface. They are grouped in pairs and checked for odd parity.

The even bit of a DCUEN[i+1:i] signal pair has positive polarity (enabled), and the odd bit has negative polarity (disabled).

A pair value of:

- `0b01` means the respective debug control is enabled.
- `0b10` means the respective debug control is disabled.
- `0b00` or `0b11` is illegal, and behaves as disabled and sets an alarm signal.

The even bit of a LCM_DCU_FORCE_DISABLE.FD[i+1:i] pair has positive polarity (force disabled), and the odd bit has negative polarity (force enabled).

A pair value of:

- `0b01` means the respective debug control is force disabled.
- `0b10` means the respective debug control value is determined by the DCUEN[i+1:i] signal pair.
- `0b00` or `0b11` is illegal, and behaves as force disabled and sets an alarm signal.

Warning

The following must comply with the rules stated in the Table 5-2: DCUEN[21:0] rules on page 65:

- Software setup of the DCU_EN<j> registers. For more information, see *Arm® Lifecycle Manager (LCM) Specification*.
- LCM_DCU_FORCE_DISABLE register
- Hardware defaults of the LCM_DCU_FORCE_DISABLE_INIT

Violating any of the pair rules results in an alarm.

Table 5-2: DCUEN[21:0] rules on page 65 shows the DCUEN[21:0] rules. In the table:

- FD[i] refers to the LCM_DCU_FORCE_DISABLE.FD[i].
- Debug Alarm and Control column describes how an alarm and a control are generated for each pair of DCUEN[21:0] bits.

Table 5-2: DCUEN[21:0] rules

LCM signal name	LCM signal use	FD	FD bit use	Debug Alarm and Control
DCUEN[0]	Positive polarity value	FD[0]	Positive polarity value	Alarm_1_0 = not(DCUEN[0] ^ DCUEN[1]) | not(FD[0] ^ FD[1])
DCUEN[1]	Negative polarity value	FD[1]	Negative polarity value	DBGEN = {DCUEN[0] & not(DCUEN[1])} & {not(FD[0] & not(FD[1]))} & {not(Alarm_1_0)}
DCUEN[2]	Positive polarity value	FD[2]	Positive polarity value	Alarm_3_2 = not(DCUEN[2] ^ DCUEN[3]) | not(FD[2] ^ FD[3])
DCUEN[3]	Negative polarity value	FD[3]	Negative polarity value	NIDEN = {DCUEN[2] & not(DCUEN[3])} & {not(FD[2] & not(FD[3]))} & {not(Alarm_3_2)}
DCUEN[4]	Positive polarity value	FD[4]	Positive polarity value	Alarm_5_4 = not(DCUEN[4] ^ DCUEN[5]) | not(FD[4] ^ FD[5])
DCUEN[5]	Negative polarity value	FD[5]	Negative polarity value	SPIDEN = { [DCUEN[4]]{.signal} & not(DCUEN[5])} & {not(FD[4] & not(FD[5]))} & {not(Alarm_5_4)}
DCUEN[6]	Positive polarity value	FD[6]	Positive polarity value	Alarm_7_6 = not(DCUEN[6] ^ DCUEN[7]) | not(FD[6] ^ FD[7])

LCM signal name	LCM signal use	FD	FD bit use	Debug Alarm and Control
DCUEN[7]	Negative polarity value	FD[7]	Negative polarity value	SPNIDEN = {DCUEN[6] & not(DCUEN[7])} & {not(FD[6] & not(FD[7]))} & {not(Alarm_7_6)}
DCUEN[8]	Positive polarity value	FD[8]	Positive polarity value	Alarm_9_8 = not(DCUEN[8] ^ DCUEN[9]) | not(FD[8] ^ FD[9])
DCUEN[9]	Negative polarity value	FD[9]	Negative polarity value	DAPACCEN = {DCUEN[8] & not(DCUEN[9])} & {not(FD[8] & not(FD[9]))} & {not(Alarm_9_8)}
DCUEN[10]	Positive polarity value	FD[10]	Positive polarity value	Alarm_11_10 = not(DCUEN[10] ^ DCUEN[11]) | not(FD[10] ^ FD[11])
DCUEN[11]	Negative polarity value	FD[11]	Negative polarity value	DAPDSSACCEN {DCUEN[10] & not(DCUEN[11])} & {not(FD[10] & not(FD[11]))} & {not(Alarm_11_10)}
DCUEN[12]	Positive polarity value	FD[12]	Positive polarity value	Alarm_13_12 = not(DCUEN[12] ^ DCUEN[13]) | not(FD[12] ^ FD[13])
DCUEN[13]	Negative polarity value	FD[13]	Negative polarity value	SYSDSSACCEN0 = {DCUEN[12] & not(DCUEN[13])} & {not(FD[12] & not(FD[13]))} & {not(Alarm_13_12)}
DCUEN[14]	Positive polarity value	FD[14]	Positive polarity value	Alarm_15_14 = not(DCUEN[14] ^ DCUEN[15]) | not(FD[14] ^ FD[15])
DCUEN[15]	Negative polarity value	FD[15]	Negative polarity value	SYSDSSACCENX = {DCUEN[14] & not(DCUEN[15])} & {not(FD[14] & not(FD[15]))} & {not(Alarm_15_14)}
DCUEN[16]	Positive polarity value	FD[16]	Positive polarity value	Alarm_17_16 = not(DCUEN[16] ^ DCUEN[17]) | not(FD[16] ^ FD[17])
DCUEN[17]	Negative polarity value	FD[17]	Negative polarity value	SYSDSSACCEN1 = {DCUEN[16] & not(DCUEN[17])} & {not(FD[16] & not(FD[17]))} & {not(Alarm_17_16)}
DCUEN[18]	Positive polarity value	FD[18]	Positive polarity value	Alarm_19_18 = not(DCUEN[18] ^ DCUEN[19]) | not(FD[18] ^ FD[19])
DCUEN[19]	Negative polarity value	FD[19]	Negative polarity value	SYSDSSACCEN2 = {DCUEN[18] & not(DCUEN[19])} & {not(FD[18] & not(FD[19]))} & {not(Alarm_19_18)}
DCUEN[20]	Positive polarity value	FD[20]	Positive polarity value	Alarm_21_20 = not(DCUEN[20] ^ DCUEN[21]) | not(FD[20] ^ FD[21])
DCUEN[21]	Negative polarity value	FD[21]	Negative polarity value	SYSDSSACCEN3 = {DCUEN[20] & not(DCUEN[21])} & {not(FD[20] & not(FD[21]))} & {not(Alarm_21_20)}

All the above alarm signals are OR-ed together to generate the debug_signals_security_checker_err signal. This signal is then OR-ed with the FATALERR signal from the Lifecycle Manager Lifecycle Indication interface. The two OR-ed signals are used to generate an alarm input at the SAM.

See Security Alarm Manager incoming events allocation for associated SAM events.

DCUEN[31:22]

The DCUEN[31:22] signals and the LCM_DCU_FORCE_DISABLE[31:22] system register bits are individually used and do not generate alarms.

The DCUEN[31:22] signals are force disabled by the respective LCM_DCU_FORCE_DISABLE.FD[31:22] register bit signals. See Table 5-3: DCUEN[31:22] rules on page 67.

In the table, FD[i] refers to LCM_DCU_FORCE_DISABLE.FD[i].

Table 5-3: DCUEN[31:22] rules

LCM signal name	LCM signal use	FD	FD bit use	Debug Control
DCUEN[31:22]	Reserved	FD[31:22]	Reserved	Reserved for internal usage (that is, BIST enable, ECOs). The use of each of DCUEN[31:22] is force disabled as follows: • DCUEN[i] & not(FD[i]) for i=22 to 31.

DCUEN[127:32]

These DCU controls are connected to Lifecycle Manager Debug Control Unit interface and do not have corresponding force disable mechanism present.

5.11 Key Management Unit

The CRSAS Ma2 uses a Key Management Unit (KMU) when LCM_KMU_SAM_PRESENT = 1. The KMU is a centralised function that stores symmetric key material for the distributed hardware countermeasures and for the use of the software with different crypto devices.

The KMU is a Secure only accessible device. Non-secure software must call Secure software to set or use a key. After a key slot is loaded, Secure software can lock it to prevent accidental reload or malicious override.

The KMU implements hardware key slots and software key slots. The keys of the hardware key slots can only be set through a private APB3 subordinate hardware keys port that connects point to point using **IMPLEMENTATION DEFINED** address to the LCM. A software or other hardware entity cannot write to this private port. When the LCM populates the hardware key slots, the software can start using them in the same way as it uses a locked software key slot.

For more information on the KMU, see the *Arm® Key Management Unit (KMU) Specification*. The *Arm® Key Management Unit (KMU) Specification* and Key Management Unit related configurations define the configuration options of the KMU.

The KMU interfaces integrated in the CRSAS Ma2 are as follows:

- APB3 subordinate interface to program the KMU registers:
  - This interface is PPC protected, always Secure access only, and the privileged/unprivileged access is configurable through the PERIPHSPPPC1.
  - Sparse writes are not allowed and must result in error response and memory intact.
  - The programming base address is defined in the Peripheral region.
- The private APB3 subordinate hardware keys port:
  - This interface is used by the LCM hardware to load hardware keys to the hardware key slots of the KMU. This interface is connected to the LCM in a point-to-point fashion that it is not accessible to software or any other hardware.
  - The memory map exposed in the private APB3 subordinate hardware keys port is LCM IMPLEMENTATION SPECIFIC. The only registers in the KMU that are accessible through this interface are the hardware key slots.

- AHB5 manager interface:
  - This interface is used by the KMU to export keys to the target crypto devices.
  - The transaction attributes that the KMU provides through this interface are Secure, and the privilege level is configurable through the PERIPHSPPPC1.
  - Additional transaction attributes that the KMU provides through this interface are defined in *Arm® Key Management Unit (KMU) Specification*.
  - This interface is connected to the top level as a dedicated Key Management Unit interface.
- Parity Error interface:
  - This interface is used by the KMU to generate alarm on detection of internal parity error.
  - This signal is connected to the SAM of the subsystem.
- Interrupt interface:
  - This interface is used by the KMU to interrupt the software.
  - This is connected to system Interrupts.
- KMU clock is synchronous to the MGMTSYSCLK clock.
- KMU Resides in the PD_MGMT power domain.
- KMU reset is primarily nCOLDRESETMGMT. However, during Secure Asset Provisioning Flow, when the LCMRSTREQ is asserted, the nWARMRESETMGMT also resets the KMU. For more information, see Reset infrastructure.

The context of the KMU must be saved and restored (retained) over HIBERNATION0. All signals that are driving PD_AON logic have to be latched and hold during HIBERNATION0.

5.12 Security Alarm Manager

The CRSAS Ma2 defines Security Alarm Manager (SAM). The SAM provides means to apply the programed response to security events detection.

The SAM and its interfaces only exist when LCM_KMU_SAM_PRESENT = 1.

For more information regarding the SAM, see *Arm® Security Alarm Manager (SAM) Specification*.

The configuration options of the SAM are defined in Security Alarm Manager related configurations and *Arm® Security Alarm Manager (SAM) Specification*.

The following SAM interfaces are integrated in the CRSAS Ma2 as follows:

- APB3 subordinate interface to program the SAM registers:
  - This interface is PPC protected, always Secure access only, and the privileged/unprivileged access is configurable. See PERIPHSPPPC1.
  - Sparse writes are not allowed and must result in error response and memory intact.
  - The programming base address is defined in the Peripheral region.
- Input Events interface:

    - This interface is available in the Sensor Alarm interface.
    - Supports both internal and external events. The integrator also connects its digital and analog attack detection sensors to these inputs.
- The External Sensors Ready input signal:
    - This input is available in the Sensor Alarm interface. The integrator must set this signal when all the external sensors are stable after power on to avoid false alarms.
- Platform-specific event logging inputs:
    - The SAM hosts platform-specific event logging registers for sources which provide many event signals, for example, Processors DCLS and RAS status signals or memory ECC. These platform-specific event logging inputs and the relevant registers are not used by the CRSAS Ma2 and not required for PSA Level 2.
- Status interface:
    - This interface is available in the Sensor Alarm interface.
    - These output signals indicate the detected events.
    - Each signal indicates, when set, that its respective sensor reported alarm condition.
- Response Action interface:
    - This interface is available in the Sensor Alarm interface.
    - Each signal indicates, when set, a response action to be taken by the system. The integrator connects each of the response signals to a mitigation action.
    - The recommended actions are detailed in Security Alarm Manager incoming events allocation.
- SAM configuration done interface:
- This interface is available in the Sensor Alarm interface.
    - SAM clock is synchronous to AONCLK
    - SAM resides in the PD_AON power domain
    - SAM resets are the nCOLDRESETAON and the nPORESETAON. see Reset infrastructure.

5.12.1 Security Alarm Manager incoming events allocation

The SAM supports various internal and external alarm events.

Table 5-4: SAM incoming events allocation on page 70 defines the allocation of these events, the recommended default behaviours, and the recommended post software configuration actions.

For more information on SAM-specific event sources, see *Arm® Security Alarm Manager (SAM) Specification*.

Note

For the Event Response Action Routing enable the SAMRESETACTION[63:0] in the Security Alarm Manager related configurations.

Available event response actions are defined in Security Alarm Manager Response Action Settings.

Table 5-4: SAM incoming events allocation

Assigned SAM Event Index	Event Source	Recommended default behaviour out of reset	Recommended Event Response Action Routing after software configuration.
0	SAM internal configuration integrity checker	1 – Enable routing the event to its default response action	0 – Cold Reset the system. This is the same as the default response action defined in *Arm® Security Alarm Manager (SAM) Specification*.
1	SAM integrated watchdog counter	0 – Do not route. Wait for SAM configuration.	2 - NMI. See Interrupts
2	Combined LCM fatal frror and DCUEN security checker error	1 – Enable routing the event to its default response action	0 – Cold Reset the system. This is the same as the default response action defined in *Arm® Security Alarm Manager (SAM) Specification*.
3-5	Reserved	0 – Do not route.	0 - Not used by the CRSAS Ma2
6	KMU internal parity error indication	0 – Do not route. Wait for SAM configuration	IMPLEMENTATION DEFINED
7-22	Reserved	0 – Do not route.	0 - Not used by the CRSAS Ma2
23	SAM internal duplication error indication	0 – Do not route. Wait for SAM configuration	IMPLEMENTATION DEFINED
24-47	Reserved	0 – Do not route.	0 - Not used by the CRSAS Ma2
48	External sensor event 0 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
49	External sensor event 1 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
50	External sensor event 2 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
51	External sensor event 3 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
52	External sensor event 4 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
53	External sensor event 5 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
54	External sensor event 6 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
55	External sensor event 7 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
56	External sensor event 8 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
57	External sensor event 9 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
58	External sensor event 10 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED

Assigned SAM Event Index	Event Source	Recommended default behaviour out of reset	Recommended Event Response Action Routing after software configuration.
59	External sensor event 11 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
60	External sensor event 12 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
61	External sensor event 13 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
62	External sensor event 14 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED
63	External sensor event 15 on Sensor Alarm interface	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED

5.12.2 Security Alarm Manager Response Action settings

The SAM supports configurable responses to alarm events.

The following table defines the required responses to alarm events. Response actions are also available in Sensor Alarm Interface to be use outside the system.

Table 5-5: SAM response action settings

Response Action ID	Action Applied	Comments
0	Cold reset	Response action ID 0 resets the SAM and LCM when it detects errors during boot to allow a retry. SAM supports this action to operate out of reset immediately. The corresponding output on the Sensor Alarm interface may be used by the SoC level to reset the entire SoC. This is done to minimise the risk of having unhandled pending transactions at system level when the reset occurs.
1	Warm reset	When this response action is taken, a controlled Warm reset entry sequence is triggered. All P-Channel enters WARM_RST state before reset assertion. This sequence is expected to drain all outstanding transactions, and reset the system in isolation without leaving pending transactions stuck in the system.
2	NMI	Triggers internal NMI interrupt. Must be cleared when the software handles the NMI. System level NMI routing must be set up in the NMI_ENABLE register.
3	Critical Severity Fault Interrupt	Connected to interrupt bit IRQ[30]. Must be cleared in SAM when the software handles the interrupt.
4	Severity Fault Interrupt	Connected to interrupt bit IRQ[31]. Must be cleared in SAM when the software handles the interrupt.
{5-7}	IMPLEMENTATION DEFINED	This SAM action can be used by the SoC integrator or be unconnected.

5.13 Power control infrastructure

The CRSAS Ma2 supports multiple possible power infrastructure levels.

These infrastructure levels are:

PILEVEL = 0

Basic Level power infrastructure

PILEVEL = 1

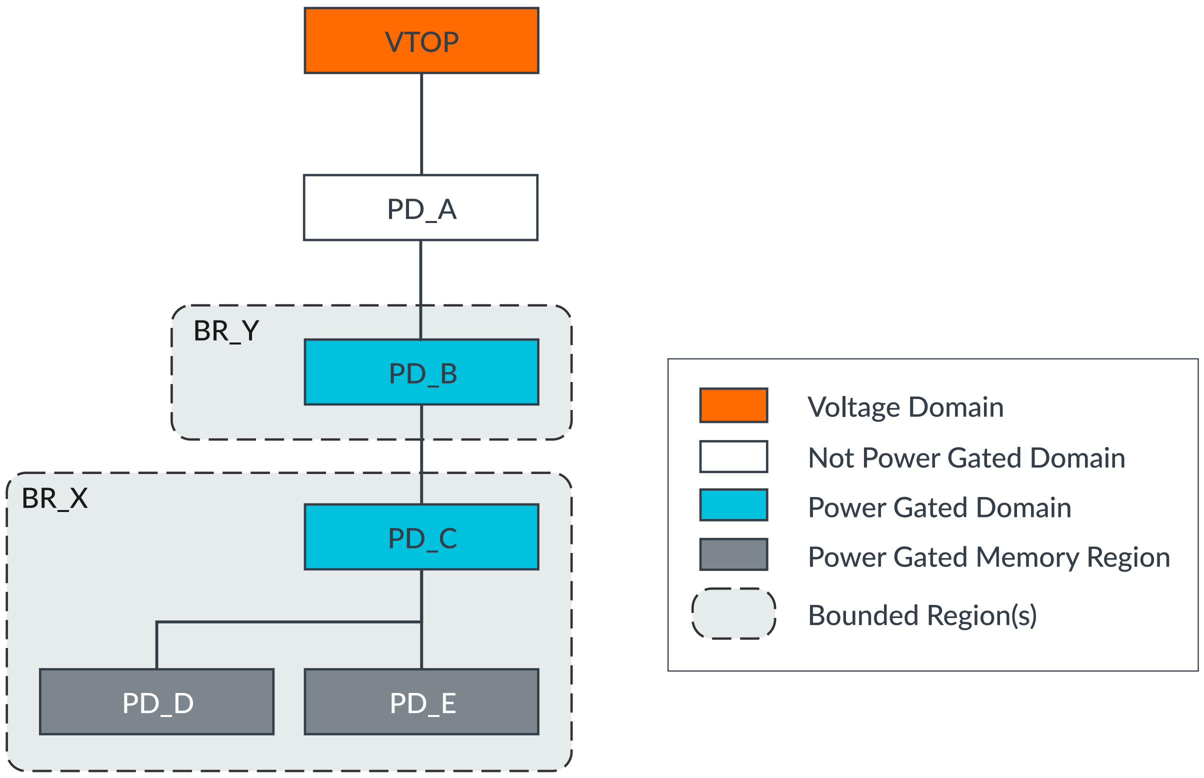
Intermediate Level power infrastructure

PILEVEL = 2

Advanced Level power infrastructure

Figure 5-7: Example power hierarchy on page 72 shows a simple power hierarchy diagram that is used to describe the power region relationships.

Figure 5-7: Example power hierarchy



In Figure 5-7: Example power hierarchy on page 72 each rectangular block represents either a voltage domain or a power domain. If a line connects two blocks where one is above another, it indicates that the domain below is within the hierarchy of the domain above. For example, PD_A region is within the VTOP voltage domain, PD_B is within PD_A, PD_C is within PD_B, and PD_D and PD_E is within PD_C. A block with dotted line indicates that the existence of the region is configuration-dependent.

Typically, when a higher-level region has several regions below it, before the hgher-region can enter a lower power state the lower-level regions must:

- Already be in a lower power state

- Transition to the lower power state at the same time

A rounded, dotted, bounding box over one or several regions indicates that their power states are controlled collectively. These are called Bounded Regions. For example, PD_C, PD_D and PD_E regions are in the bounded region BR_X. A bounded region is often controlled using a single PPU. PPUs are complemented by LPI infrastructure components. This is done to bring together the quiescence status and control of IP blocks. This happens primarily within the power domains that are controlled by each PPU.

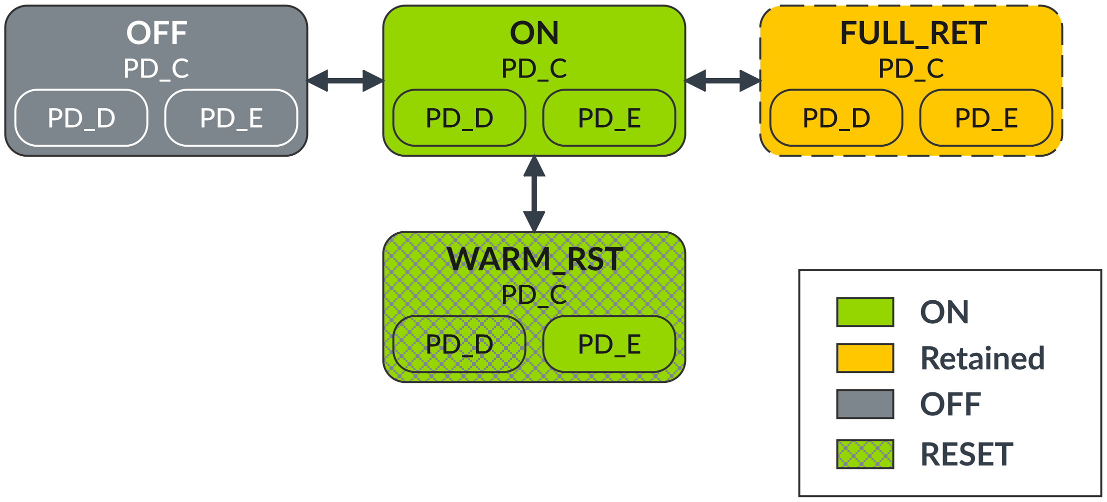
The power modes of a bounded region are represented using a state transition diagram that shows all the modes and the supported transitions between them. For example, Figure 5-8: Example Power Mode Transition Diagram of three bounded power regions, PD_C, PD_D and PD_E on page 74 shows a simple four modes transition diagram for a bounded region with three power regions: PD_C, PD_D, and PD_E.

Each state is represented by a box that represents the higher hierarchy region power state. Internal boxes represent lower hierarchy region power states. A name is given to each power mode in bold. For example, Figure 5-8: Example Power Mode Transition Diagram of three bounded power regions, PD_C, PD_D and PD_E on page 74 only has four bounded region power modes:

- OFF
- ON
- WARM_RST
- FULL_RET

The different colors indicate the power state of each region. The patterned fill of a region indicates a bounded region mode. This mode is specifically entered to prepare for the application of reset for all the registers within the corresponding Warm reset domain. The PD_E is never reset because it is a memory power domain. The dashed lined block indicates that the mode is optional.

Figure 5-8: Example Power Mode Transition Diagram of three bounded power regions, PD_C, PD_D and PD_E



Note

These mode diagrams are very similar to the PPU power mode transitions. However, the diagrams here and their modes do not indicate that a power mode in the PPU with the same name is used. However, it is likely that they match, and it is **IMPLEMENTATION DEFINED** how these are mapped to the PPU modes.

Along with the power modes, each bounded region can also support operating modes. Operating modes allow additional features within the bounded region to be controlled. For example, a bounded region can have four operating modes that are related to the four possible combination of "enabled" states of the two SRAM banks within the bounded region. For more details of operating modes, see *Arm® Power Policy Unit Architecture Specification*.

A Power Dependency Control Matrix (PDCM) is also defined as a table for each PILEVEL. The PDCM is used to describe and control how each power domain affects another and the lowest power state each domain is allowed to enter.

For each PILEVEL, a System Level Power State table is provided. The table defines and describes several high-level power states of the system in relation of the power domains states. Each row of table describes: - a defined System Level Power state - the supported power states of the power domains - the supported power states of other system states like clock availability and voltage supply

Combinations of power domain states that are not supported in the table are not supported by the system.

5.13.1 Advanced level power infrastructure

The advanced level power infrastructure specifies a power control infrastructure that provides additional functionality. This is to help an implementation achieve very low power at the lowest System Power State. This is achieved by providing more voltage and power domains.

The system at PILEVEL = 2 supports two voltage domains:

VAON

The always ON voltage domain.

This voltage domain is intended only to drive the always ON logic and therefore only includes the PD_AON domain. The domain must always be always ON.

This domain is separated from the VSYS. This is done to allow always ON logic to be implemented using a different target process technology in order to reduce always ON power.

VSYS

The system voltage domain.

This voltage domain implements all other power domains in the system. VSYS, if required, can be powered off in the lowest power state. However all retention requirement of any host logic host within it must be met even during the lowest power state.

5.13.1.1 Power hierarchy

The system at PILEVEL = 2 supports power domains that are hierarchical.

The system supports the following power domains:

- PA_AON
- PD_DEBUG
- PD_MGMT
- PD_SYS
- PD_VMR<*i*>, where *i* is 0 to NUMVMBANK-1, and does not exist if NUMVMBANK = 0
- PD_NPU<*m*>, does not exist if NUMNPU = 0
- PD_CPU<*n*>
- PD_CPU<*n*>EPU
- PD_CPU<*n*>RAM
- PD_CPU<*n*>TCM

The following figure shows the voltage and power domain hierarchy.

Figure 5-9: Voltage and Power domain hierarchy of a subsystem with PILEVEL = 2

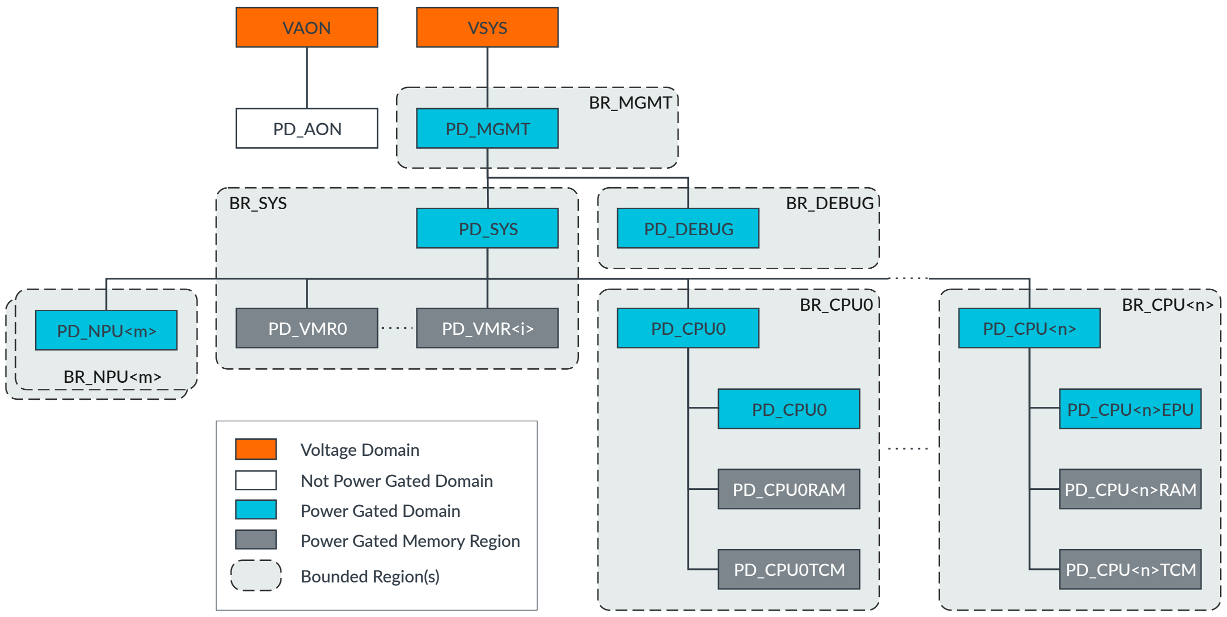


Figure 5-9: Voltage and Power domain hierarchy of a subsystem with PILEVEL = 2 on page 76 shows several bounded regions. Each of these bounded regions are controlled by a single PPU. The bounded regions and their controlling PPU are as follows:

BR_MGMT

This is controlled by the single PPU, the MGMT_PPU. The power domain in the BR_MGMT is PD_MGMT.

BR_DEBUG

This is controlled by the single PPU, the DEBUG_PPU. The power domain in BR_DEBUG is PD_DEBUG. This is a combined power gated logic and memory domain for all debug logic in the system.

BR_SYS

This is controlled by the single PPU, the SYS_PPU. The power domains in the BR_SYS are:

- PD_SYS
- PD_VMR<i> one for each VM<i>, where i is 0 to NUMVMBANK – 1. This power domain does not exist when NUMVMBANK = 0.

BR_CPU<*n*>

Each are controlled by the single PPU, the CPU<*n*>_PPU. The power domains in BR_CPU<*n*> are:

- PD_CPU<*n*>
- PD_CPU<*n*>EPU
- PD_CPU<*n*>RAM
- PD_CPU<*n*>TCM

BR_NPU<m>

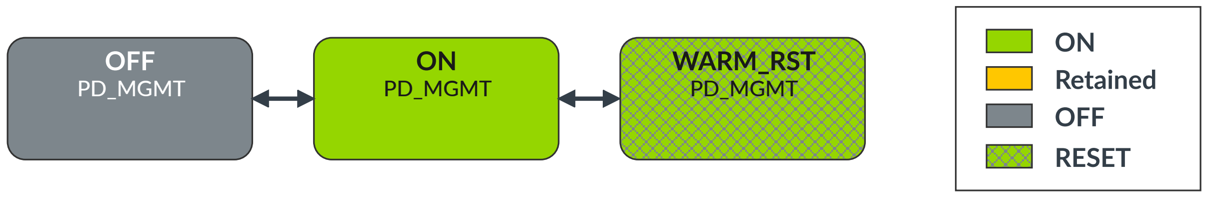
Each bounded region is controlled by the single PPU, the NPU<m>_PPU. The power domain in BR_NPU<m> is PD_NPU<m>

5.13.1.2 BR_MGMT power modes

The BR_MGMT supports several power modes.

The following figure shows the supported power modes:

Figure 5-10: BR_MGMT power mode transition diagram



On Cold reset the PD_MGMT automatically transitions to ON state and enters OFF state eventually. If a Warm reset is requested, the bounded region power state enters the WARM_RST when the domain is idle and ready for reset. The domain dynamically enters the OFF state.

The domain always either enters or stays in the ON state on the following conditions:

- Any of the other power domains that lie within the PD_MGMT is entering or is already in the ON state.
- MGMTWAKEUP Q-Channel Device interface for the PD_MGMT is driven to request to leave the OFF state.

The BR_MGMT power modes do not specifically support the FUL_RET state. However, some registers that reside in the PD_MGMT domain may need to be retained when PD_MGMT is OFF, and it is **IMPLEMENTATION DEFINED** how this is achieved.

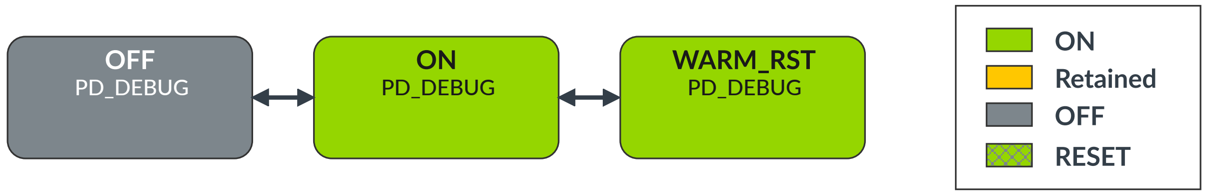
5.13.1.3 BR_DEBUG power modes

On Cold reset, the PD_DEBUG automatically transitions to ON state. If the debug system is not required to stay ON it eventually enters OFF state. It is woken either through the Power Control Wakeup Q-Channel Device interface of the PD_DEBUG, or through a bus access targeting its shared debug domain.

The WARM_RST state is entered to bring the debug domain into an idle state, so that the main system can be Warm reset cleanly. The WARM_RST state is entered, even though the debug domain itself is not being Warm reset at the same time.

The following figure shows the power modes that the BR_DEBUG supports.

Figure 5-11: BR_DEBUG power mode transition diagram

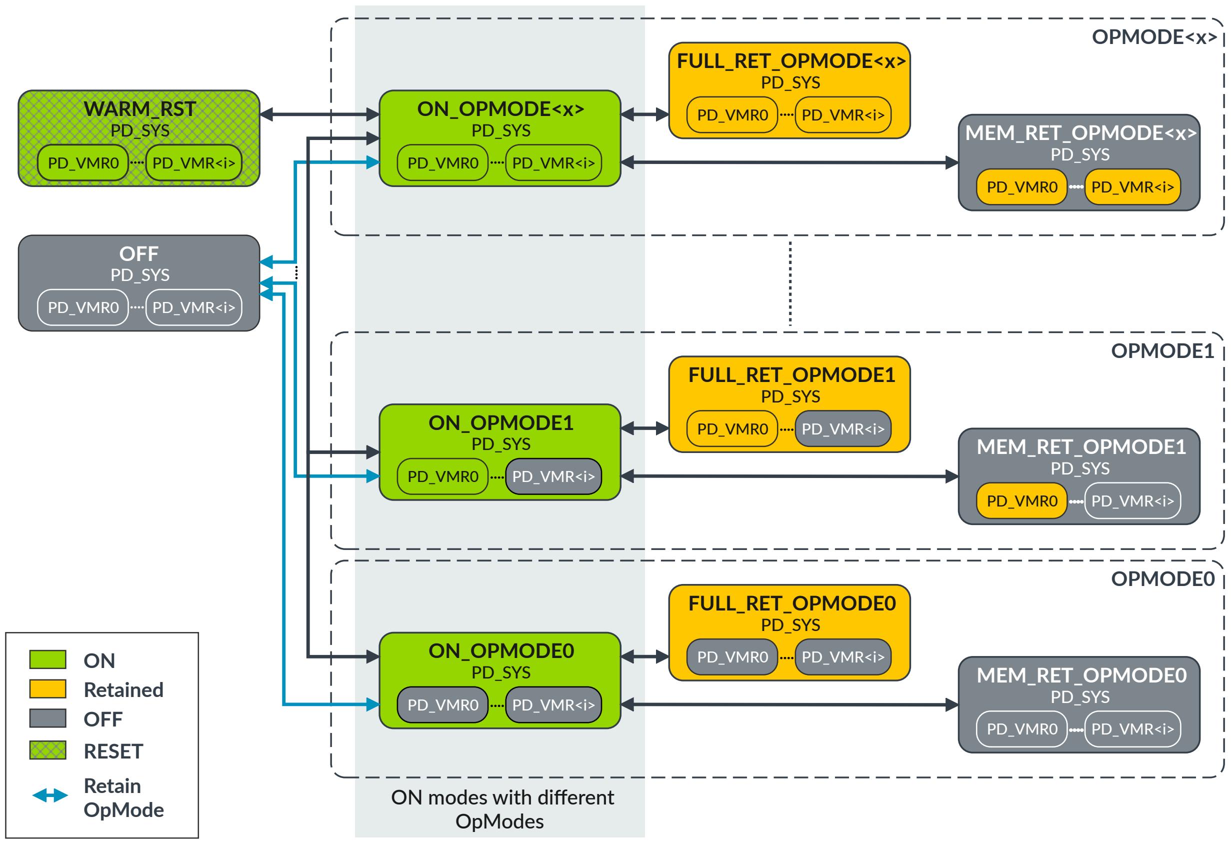


5.13.1.4 BR_SYS power modes

The BR_SYS supports several power modes.

The following figure shows the supported power modes:

Figure 5-12: BR_SYS power mode transition diagram



BR_SYS power modes have $2^{NUMVMBANK}$ different operating modes, which can be: 1, 2, 4, 8, or 16. This is encoded as a binary value of up to four bits, with each bit for a PD_VMR<*i*>. Each bit indicates a specific PD_VMR<*i*> to be enabled or disabled.

When NUMVMBANK = 0, the BR_SYS power modes only have one operating mode. This means that a PPU supporting the BR_SYS does not need to support operating modes. When a PD_VMR<*i*> is disabled, its lower power mode is always in the OFF state. Other than the OFF or WARM_RST states, it is only possible to move between the operating modes while in one of the ON modes.

For example, for a system with NUMVMBANK = 2, the operating modes are:

OPMODE0

The PD_VMR0 and the PD_VMR1 are both disabled.

OPMODE1

The PD_VMR0 is enabled and the PD_VMR1 is disabled.

OPMODE2

The PD_VMR0 is disabled and the PD_VMR1 is enabled.

OPMODE3

The PD_VMR0 and the PD_VMR1 are both enabled.

On Cold reset, the system enters the ON_OPMODE&lt;NUMVMBANK-1>. In this power mode, all the PD_VMR<*i*> are enabled.

If a Warm reset is requested, the bounded region transitions to the WARM_RST through other power modes when the domain is idle and ready for reset.

Note

The DMA includes a minimum power register in its programmers model. If the DMA is included in the system (NUMDMA &gt; 0), it can prevent the PD_SYS from transitioning to a power state that is lower than what its internal minimum power register allows.

5.13.1.4.1 Controlling the PD_VMR<i> power mode

To configure the required low-power state of each PD_VMR<*i*>, the software must configure the register field PDCM_PD_VMR<*i*>_SENSE.MIN_PWR_STATE as follows:

`0b00` to set the low-power state of the PD_VMR<*i*> to OFF

The PD_VMR<*i*> only ever transitions between the ON and OFF states. It is never in retention. This means that in low-power state, all state in the VM<*i*> is lost.

With this setting, when the PD_SYS is ON and the BR_SYS is in one of the ON_OPMODE modes that currently has PD_VMR<*i*> ON, it transitions to another ON_OPMODE. In this ON_OPMODE the PD_VMR<*i*> is OFF automatically when the PD_VMR<*i*> domain is idle. Therefore, expect to see the PD_VMR<*i*> turn off quickly after the PDCM_PD_VMR<*i*>_SENSE.MIN_PWR_STATE is set to `0b00`.

When the PD_VMR<*i*> is OFF, it can only be returned to ON by an access on the bus targeting the PD_VMR<*i*>. Following that, when the PD_VMR<*i*> is idle again, it turns off again. To avoid this, configure the PDCM_PD_VMR<*i*>_SENSE.MIN_PWR_STATE to a nonzero value before accessing the VM<*i*>.

`0b01` to set the low-power state of the PD_VMR<*i*> to retention

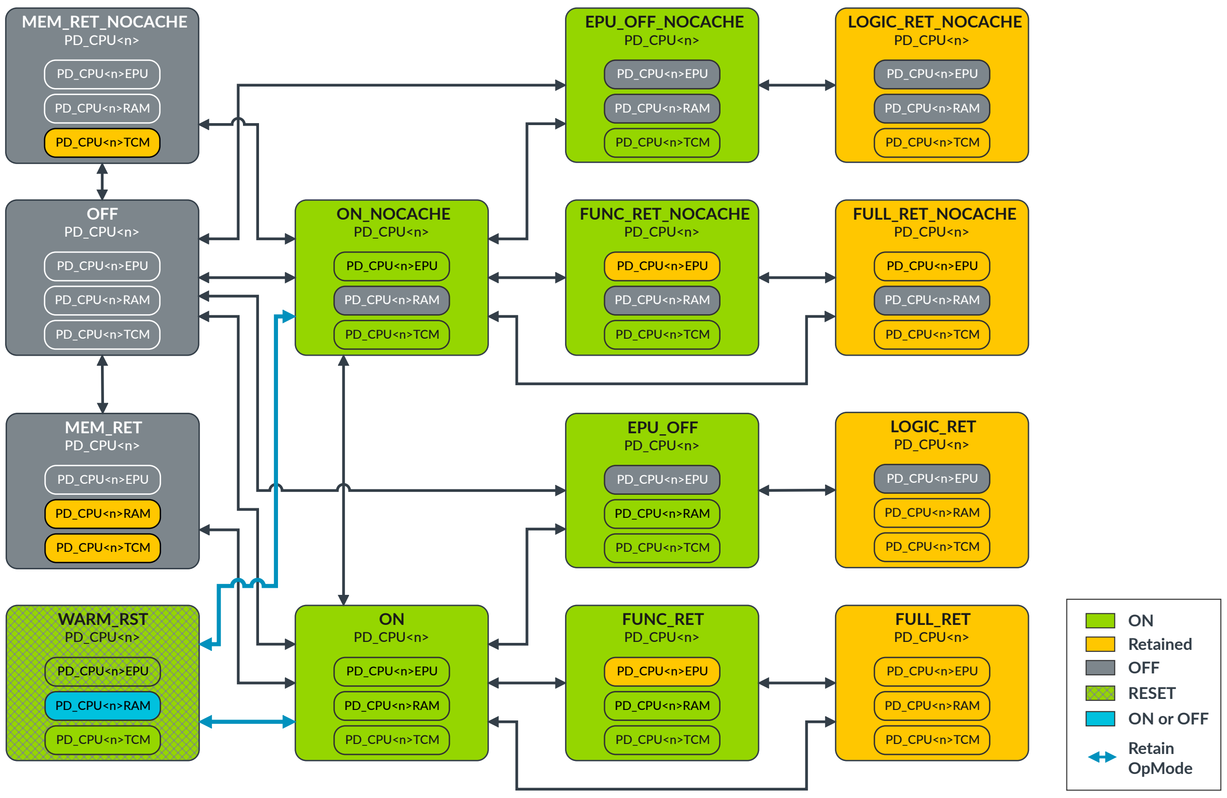
The PD_VMR<*i*> only transitions between the ON and retention state, and never enters the OFF state. This means that in low-power state, all register states in the VM<*i*> are retained. Therefore, for the BR_SYS, it never transitions to a power mode that has the PD_VMR<*i*> turned off. To place the the PD_VMR<*i*> into retention, the BR_SYS power modes have to enter one of the FULL_RET_OPMODE modes, or a MEM_RET_OPMODE mode where PD_VMR<*i*> is in retention. This means that it is not possible to place a PD_VMR into retention while keeping PD_SYS ON.

5.13.1.5 BR_CPU<n> power modes

The BR_CPU<n> supports several power modes.

The following figure shows the supported power modes:

Figure 5-13: BR_CPU<n> power mode transition diagram



On Cold reset, the bounded region enters the EPU_OFF_NOCACHE power mode.

Figure 5-13: BR_CPU<n> power mode transition diagram on page 80 is like that supported by Cortex-M55 and Cortex-M85 cores, expect that an additional PD_CPU<n>TCM power region is added.

The PD_CPU<n>TCM power state always matches that of the PD_CPU<n>, except in the MEM_RET or MEM_RET_NOCACHE states. In these states, it is to be retained. The choice from the ON_NOCACHE to either the MEM_RET_NOCACHE or the OFF is determined by the local TCM minimum power state register configuration of the processor, the CPUPWRCFG.TCM_MIN_PWR_STATE.

If a Warm reset is requested, the bounded region transitions to the WARM_RST through other power modes when the domain is idle and ready for reset.

5.13.1.5.1 Controlling the PD_CPU<n>RAM power states

The BR_CPU<n> bounded region uses operating modes to support the ability to turn on or off the cache RAMs in modes other than the OFF mode. Power modes with cache RAMs disabled, called the NOCACHE operating modes, are suffixed with NOCACHE. Power modes with cache RAMs enabled, called the CACHE operating modes, are without the NOCACHE suffix. To select the use of the NONCACHE operating modes, the following registers must be configured:

- Set CPDLPSTATE.RLPSTATE register in the processor to OFF which is `0b11`
- Set MSCR.DCACTIVE register in the processor to `0b0` to disable the Data Cache
- Set MSCR.ICACTIVE register in the processor to `0b0` to disable the Instruction Cache

Transitions between the two operating modes can only occur between the ON and ON_NOCACHE power modes.

When in the NOCACHE operating modes, the PD_CPU<n>RAM is always turned off. When operating in the CACHE operating modes, the PD_CPU<n>RAM enters RAM retention when entering the MEM_RET, LOGIC_RET, or FULL_RET power modes.

5.13.1.5.2 Controlling the PD_CPU<n>EPU power states

Other than the WARM_RST state, the BR_CPU<n> bounded region provides the ability for the PD_CPU<n>EPU to enter a lower power state independently, as long as the PD_CPU<n> is ON. To control whether the PD_CPU<n>EPU remains ON, or be allowed to enter the Retention (RET) or OFF state, the software can use the register CPDLPSTATE.ELPSTATE in conjunction with the CPPWR.SU10 in the processor as follows:

RET

To allow the PD_CPU<n>EPU to enter the Retention state where the BR_CPU<n> never enters the EPU_OFF, LOGIC_RET EPU_OFF_NOCACHE, LOGIC_RET_NOCACHE, MEM_RET, MEM_RET_NOCACHE, and OFF states, set either:

- CPDLPSTATE.ELPSTATE to RET, `0b10`

- CPDLPSTATE.ELPSTATE to OFF, `0b11`. Also set the CPPWR.SU10 to `0b0`. Retention is only entered when the CPU<n> is sleeping using WFI or WFE.

OFF

To allow the PD_CPU<n>EPU to enter the OFF state where the BR_CPU<n> never requests FUNC_RET, FUNC_RET_NOCACHE, FULL_RET_NOCACHE, and FULL_RET states:

- Set the CPDLPSTATE.ELPSTATE to OFF, `0b11`. Also set the CPPWR.SU10 to `0b1`.

ON

To keep the PD_CPU<n>EPU in the ON, or context retained state where the BR_CPU<n> never requests the EPU_OFF, LOGIC_RET EPU_OFF_NOCACHE, LOGIC_RET_NOCACHE, MEM_RET, MEM_RET_NOCACHE, FUNC_RET, FUNC_RET_NOCACHE, and OFF states:

- Set the CPDLPSTATE.ELPSTATE to ON, which is `0b00` or `0b01`.

Note

Depending on the processor core and the system implementation, other conditions may be required to reach the desired state.

For more information on these conditions, see the power control of the processors in *Arm® Cortex®-M55 Processor Technical Reference Manual*, *Arm® Cortex®-M55 Processor Integration and Implementation Manual*, and *Arm® Cortex®-M85 Processor Integration and Implementation Manual*.

For more information on CPPWR, see *Arm®v8-M Architecture Reference Manual*.

5.13.1.5.3 Controlling the PD_CPU<n> power states

For the PD_CPU<n> to enter a lower power state, the software on the CPU<n> must first configure its CPDLPSTATE.CLPSTATE register. This register defines what power state the CPU<n> can enter when in a sleep state.

The possible power states that the CPU<n> can enter when in a sleep state are as follows:

Set the CPDLPSTATE.CLPSTATE to RET

To allow the PD_CPU<n> to enter retention state only when in a sleep state. When set to RET, the BR_CPU<n> never enters the OFF, MEM_RET_NOCACHE, or MEM_RET modes. Other modes can be entered depending on the PD_CPU<n>EPU and the PD_CPU<n>RAM domain power states.

Set the CPDLPSTATE.CLPSTATE to OFF

To allow the PD_CPU<n> to enter the OFF state only when in a sleep state. When set to OFF, the BR_CPU<n> never requests the LOGIC_RET_NOCACHE and LOGIC_RET modes. Other modes can be entered if a coprocessor (CP<i>), which is included in the system, is indicating that the state cannot be lost (CPPWR.SU<i>=`0b0`). Entering other modes depends on the PD_CPU<n>EPU and the PD_CPU<n>RAM domain power states.

Set the CPDLPSTATE.CLPSTATE to ON, that is to `0b00` or `0b01`

To keep the PD_CPU<n> ON even when it is in a sleep mode. This means that the BR_CPU<n> never enters the LOGIC_RET_NOCACHE, LOGIC_RET, FULL_RET_NOCACHE, FULL_RET, OFF, MEM_RET_NOCACHE, and MEM_RET modes. Other modes can be entered depending on the PD_CPU<n>EPU and the PD_CPU<n>RAM domain power states.

Note

Depending on the processor core and the system implementation, other conditions may be required to reach the desired state. For more information on these conditions see the power control of the processors in *Arm® Cortex®-M55 Processor Technical Reference Manual*, *Arm® Cortex®-M55 Processor Integration and Implementation Manual*, and *Arm® Cortex®-M85 Processor Integration and Implementation Manual*.

For more information on CPPWR, see *Arm®v8-M Architecture Reference Manual*.

For each processor to then enter a lower power state, the processor must enter Wait For Interrupt (WFI) DeepSleep state. In the CRSAS Ma2, DeepSleep always uses a WIC. External WIC for each processor always exists, and all the External WICs reside in the PD_AON domain. Depending on the HASCPU<n>IWIC:

- If the HASCPU<n>IWIC is set to `0b0` for the CPU<n>, then the internal WIC does not exist for the CPU<n>. In this case the external WIC is always used for the CPU<n>.
- If the HASCPU<n>IWIC is set to `0b1` for the CPU<n>, then the internal WIC also exists. In this case the choice of which WIC to use is a software choice through the CPUPWRCFG.USEIWIC. For more information, see CPUPWRCFG.

Any internal WIC that exists resides in the associated PD_CPU<n> power domain.

The types of power states each processor in the CRSAS Ma2 supports with information on how to enter each are as follows:

OFF - DeepSleep state

Allows the processor to turn off, but utilises interrupts through the external WIC to wake the processor. To enter this state, the processor must do the following before entering WFI:

1. Select to use the External WIC through the CPUPWRCFG.USEIWIC register
2. Set the CPDLPSTATE.CLPSTATE of the CPU<n> to OFF
3. Enable DeepSleep

If Internal WIC is selected PD_CPU<n> is not able to turn off and remains ON.

RET - DeepSleep state

Allows the processor to enter the retention state, and utilises interrupts through the external WIC to wake the processor. To enter this state, the processor must do the following before entering WFI:

1. Select to use the External WIC through the CPUPWRCFG.USEIWIC register

2. Set CPDLPSTATE.CLPSTATE to RET
3. Enable DeepSleep

If the Internal WIC is selected instead, the PD_CPU<n> is not able to enter retention and remains ON.

ON - DeepSleep state

Allows the processor to enter a sleep state that only supports stopping the processor clock internally, including to the NVIC. It can use either the external or internal WIC to wake the processor. A selection of external or internal WIC can be performed using the CPUPWRCFG.USEIWIC register prior to entering WFI. To enter this state, the processor must do the following before entering WFI:

1. Set the CPDLPSTATE.CLPSTATE to `0b01`, ON
2. Enable DeepSleep

ON - Sleep state

Allows the processor to enter a sleep state that is still ON, keeping its NVIC clocking and running with the rest of the core clock turned off. To enter this state, before entering WFI or WFE the processor:

- Must not enable DeepSleep
- Must not set CPDLPSTATE.CLPSTATE to `0b00`, ON

ON state

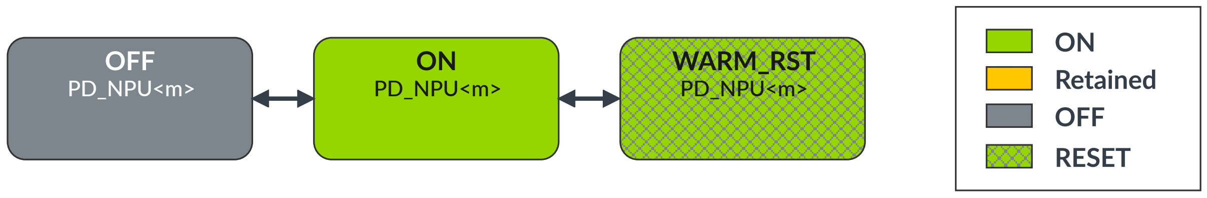
The normal running state of the processor. In this state, the EPU and the RAMs in the processor have a degree of separate control as detailed in Controlling the PD_CPU<n>RAM power states and Controlling the PD_CPU<n>EPU power states.

5.13.1.6 BR_NPU<m> power modes

The BR_NPU<m> supports several power modes.

The following figure shows supported power modes:

Figure 5-14: BR_NPU power mode transition diagram



On Cold reset, the bounded region enters the ON state.

If a Warm reset is requested, the bounded region transitions to the WARM_RST when all dependent domain is idle and ready for reset. The CMD register of the NPU can be programmed to prevent the NPU powering off by setting its power_q_enable field to 0.

For more information of this register, see *Arm® Ethos™-U55 NPU Technical Reference Manual* or *Arm® Ethos™-U65 NPU Technical Reference Manual*. After the reset, the NPU does not block power down when it enters an idle state.

5.13.1.7 Advanced power dependency control

The power dependency control implements a Power Dependency Control Matrix (DPCM) which is used to describe and control how each power domain affects another and the lowest power state each domain is allowed to enter. It allows, as much as possible, for the power control of the system to be performed primarily using dynamic power transitions. This reduces the number of software interactions needed for system management. This improves its responsiveness and contributes to further power reduction. The PDCM and all other sensitivity are defined for each power domain.

Table 5-6: Power Dependency Matrix when PILEVEL = 2 on page 86 shows an example, with PDCMQCHWIDTH of 4 and four processor cores, how each of the power domains are affected by the power state of the other domains.

The heading row of the table lists the Power Domains that are being controlled while the left column lists the Power dependency inputs. Power dependency inputs are either:

- The “ON” state of each power domain in the system. For example, the PD_MGMT_ON means that the PD_MGMT is ON when asserted.
- Expansion Power Dependency Control Matrix Q-Channel signals. The PDCMONQREQn an PDCMRETQREQn signals are driven by expansion logic from outside the subsystem This indicates a keep-up request from external power domains.

Conf

Indicates that the power domain is software configurable.

Y

Indicates that the power domain is always sensitive to the respective dependency input.

For example, the PD_SYS can be software configured to be sensitive to the ON state of the PD_SYS and all Expansion Power Control Dependency inputs. However, it is always sensitive to the ON state of all the PD_CPU<*n*> and the PD_NPU<m>.

When a power domain is sensitive to an ON dependency input, it means the following:

- When the controlled power domain is already ON and any of the dependency inputs is ON or true, then the power domain remains ON. The exception is with the PDCMRETQREQn inputs. With these inputs, if a power domain is configured to be sensitive, the power domain maintains at least in a retention state. The PDCM is primarily used to define when a power domain should not enter a lower power state. It is not designed to support powering up of any power domain.

Note

In Table 5-6: Power Dependency Matrix when PILEVEL = 2 on page 86, the following apply:

- PD_VMR<i> columns do not exist if *i* > (NUMVMBANK – 1)
- PD_CPU<*n*>_ON rows do not exist if *n* > NUMCPU
- PD_NPU<*m*>_ON rows do not exist if NUMNPU= 0
- In the cases of PDCMONQREQn[k] and PDCMRETQREQn[k], [k] is {0-<PDCMQCHWIDTH-1>}.

Table 5-6: Power Dependency Matrix when PILEVEL = 2

Power Dependency Inputs/ Power Domain	PD_MGMT	PD_SYS	PD_VMR0	PD_VMR1	PD_VMR2	PD_VMR3
PD_MGMT_ON	Conf	-	-	-	-	-
PD_SYS_ON	Y	Conf	-	-	-	-
PD_CPU0_ON	Y	Y	Conf	Conf	Conf	Conf
PD_CPU1_ON	Y	Y	Conf	Conf	Conf	Conf
PD_CPU2_ON	Y	Y	Conf	Conf	Conf	Conf
PD_CPU3_ON	Y	Y	Conf	Conf	Conf	Conf
PD_NPU0_ON	Y	Y	Conf	Conf	Conf	Conf
PD_NPU1_ON	Y	Y	Conf	Conf	Conf	Conf
PD_NPU2_ON	Y	Y	Conf	Conf	Conf	Conf
PD_NPU3_ON	Y	Y	Conf	Conf	Conf	Conf
PD_DEBUG_ON	Y	-	-	-	-	-
PDCMONQREQn[k]	Y	Conf	Conf	Conf	Conf	Conf
PDCMRETQREQn[k]	Conf	Conf	Conf	Conf	Conf	Conf

The PD_VMR<i> can also be configured to be sensitive to a power domain. For example the memory also remains ON, if both are true:

- The PD_VMR0 sensitivity is configured by the software to be sensitive to the PD_CPU0_ON
- The PD_CPU0 is ON

However, the PD_VMR0 is controlled using the same PPU as the PD_SYS. As a result of the power state of the PPU shown in BR_SYS power modes, whenever the PD_VMR0 is ON, the PD_SYS also remains in one of the ON_OPMODE states where the PD_VMR0 is ON.

Note

The PD_SYS and the PD_MGMT can be configured to be sensitive to themselves. When set up by the software to do so, each domain remains ON after it is ON.

The CRSAS Ma2 also provides a programmable register for the following power domains shown in the following table, which defines the lowest power mode that each domain can enter. The PD_DEBUG and the PD_NPU<m>, if they exist, default to a minimum power state of OFF.

Table 5-7: Power Domain's Minimum Power State, for PILEVEL = 2

Power Domain	Supported MIN_PWR_STATE for each domain
PD_MGMT	ON, OFF.
PD_SYS	ON, OFF, Retention.
PD_VMR<i>	ON, OFF, Retention.

The MIN_PWR_STATEs of the PD_MGMT, the PD_SYS, and the PD_VMR help to determine which power modes of the BR_SYS PPU can be used next when performing dynamic power transition.

For example, if all the following are true:

- The system is idle
- All dependent domains are not ON
- The BR_SYS PPU tries to enter the bounded region collectively to a low-power state
- If MIN_PWR_STATE of PD_SYS is set to Retention

Then the BR_SYS is never allowed to enter the OFF state, nor any of the MEM_RET_OPMODE<x> states. This is because the PS_SYS is not allowed to turn off. In this case, depending on the current PD_VMR<i> state, it tries to enter one of associated FULL_RET_OPMODE<x> states.

In another example, if all the following are true:

- The PDCM_PD_VMR<x>_SENSE.MIN_PWR_STATE of the PD_VMR0 is ON
- All MIN_PWR_STATE of other PD_VMRs are OFF
- Currently all the PD_VMR<i> are ON when the BR_SYS PPU is entering a low-power state

Then it transitions to ON_OPMODE1 state to turn off all the PD_VMR<i> except for the PD_VMR0. It never enters the FULL_RET_OPMODE1 nor the MEM_RET_OPMODE1.

5.13.1.8 Advanced System Level Power States

Through the relationship defined by the power dependency control matrix, the MIN_PWR_STATEs of each power domain and the processor minimum power states, we can define several System Power States that the system supports.

The supported System Power States are as follows:

SYS_OFF

All voltage and power domains are OFF.

HIBERNATION1

The VSYS voltage domain can be ON or OFF, and the system is in the lowest power state that can still be awaken from sleep. At wake, the system has to reboot.

HIBERNATION0

The VSYS voltage domain is ON. The system is in the second lowest power state, where the PD_MGMT is still ON. The system can be awaken from sleep, potentially quicker than from HIBERNATION1. At wake, the system has to reboot.

SYS_RET

The system is in retention. At wake, the system can continue to execute since no system state is lost.

SYS_ON

The system is ON.

Table 5-8: System Level Power States on page 88 lists the power states of each power domain that each of the System Power State supports or requires. See controlling PD_CPU<n> power states.

Note

In Table 5-8: System Level Power States on page 88, the following apply:

- When CPU<n> is in OFF-DeepSleep state, the TCM associated with that processor can either be:
    - OFF
    - In retention and is defined by CPUPWRCFG.TCM_MIN_PWR_STATE. In this state, the EPU must be OFF. The PD_CPU<n>RAM can be OFF permanently and is defined by CPDLPSTATE.RLPSTATE.
- PD_NPU<m> power domains only exist when NUMNPU > 0.
- PD_VMR<i> power domains only exist if NUMVMBANK > 0.

Table 5-8: System Level Power States

Power Domains	SYS_OFF	HIBERNATION1	HIBERNATION 0	SYS_RET	SYS_ON
VAON (PD_AON)	OFF	ON	ON	ON	ON
PD_MGMT	OFF	OFF	ON	ON	ON
PD_SYS	OFF	OFF	OFF	RET	ON
PD_CPU<n>	OFF	OFF - DeepSleep	OFF - DeepSleep	OFF - DeepSleep RET - DeepSleep	OFF - DeepSleep RET - DeepSleep ON - DeepSleep ON - Sleep ON
PD_NPU<m>	OFF	OFF	OFF	OFF	OFF/ON
PD_DEBUG	OFF	OFF	OFF/ON	OFF/ON	OFF/ON
PD_VMR<i>	OFF	OFF/RET	OFF/RET	OFF/RET	OFF/ON

The following observations can be made based on the table:

- When the PD_NPU<m> is ON, the PD_SYS must also be ON, and therefore the PD_MGMT is also ON.
- When waking up any of the the PD_CPU<*n*> or PD_NPU<m> power domains, if the PD_SYS is OFF or RET, it always also automatically wakes the PD_SYS domain to ON. This is because the

PD_CPU<*n*> ON and the PD_NPU<m> ON states are only supported in the SYS_ON System Power State. This means that PD_MGMT also automatically turns ON.

- In the HIBERNATION0 state the PD_MGMT is ON, and the PD_DEBUG and can be woken independently.
- In the HIBERNATION1 state the PD_MGMT is OFF. Anything that wakes the system must also wake the PD_MGMT.
- The PD_VMR<i> is tied to the PD_SYS because they belong to the bounded region BR_SYS. You are not able to wake the PD_VMR<i> independently from the PD_SYS.

An additional transient WARM_RST state exists for the system when a Warm reset is being performed. This state can only be entered from the SYS_ON state. In this state, all power domains, except for the PD_AON, temporarily enter WARM_RST mode and exit it when Warm reset is completed.

Note

When the PD_MGMT is OFF, the VSYS can also turn OFF. However, in some scenarios - depending on implementation or integration - the VSYS can remain ON if the VAON is also ON.

5.13.2 Intermediate level power infrastructure

The intermediate level power infrastructure makes some simplifications to reduce the complexity of the infrastructure. However, this simplification costs the reduction of the support provided to reduce always-on leakage power and power during HIBERNATE1 state.

The system at PILEVEL = 1 supports only one voltage domain:

VSYS

The system voltage domain. The previous VAON voltage domain is now merged into the system voltage domain. Therefore VSYS is now also considered as an always ON voltage domain.

In addition, the PD_MGMT is merged into the PD_AON domain.

5.13.2.1 Power hierarchy

The system at PILEVEL = 1 supports power domains that are hierarchical.

The system supports the following power domains:

- PD_AON
- PD_DEBUG
- PD_SYS
- PD_VMR<*i*>, does not exist if NUMVMBANK = 0
- PD_NPU<*m*>, does not exist if NUMNPU = 0

- PD_CPU<*n*>
- PD_CPU<*n*>EPU
- PD_CPU<*n*>RAM
- PD_CPU<*n*>TCM

Figure 5-15: Voltage and Power domain hierarchy of a subsystem with PILEVEL = 1 on page 90 shows the voltage and power domain hierarchy. In the diagram, the PD_MGMT is merged and part of the PD_AON.

Figure 5-15: Voltage and Power domain hierarchy of a subsystem with PILEVEL = 1

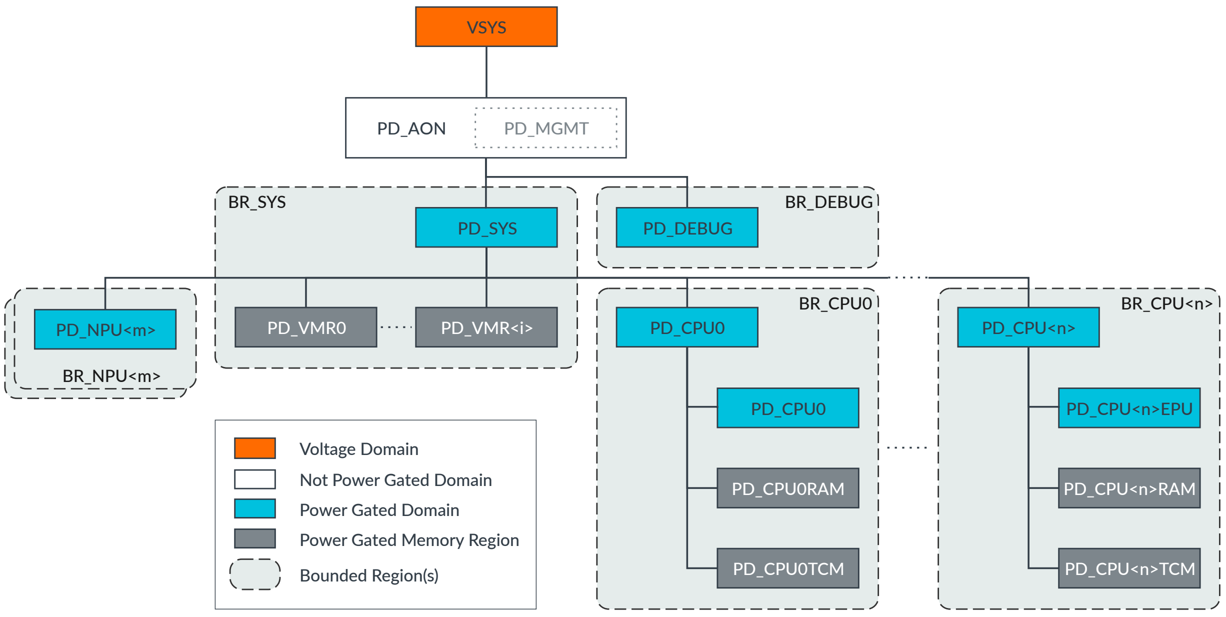


Figure 5-15: Voltage and Power domain hierarchy of a subsystem with PILEVEL = 1 on page 90 shows several bounded power regions:

BR_DEBUG

This is controlled by a single PPU, the DEBUG_PPU. The power domain in the BR_DEBUG is the PD_DEBUG. This is a combined power gated logic and memory domain for all debug logic in the system.

BR_SYS

This is controlled by a single PPU, the SYS_PPU. The power domains in the BR_SYS are:

- PD_SYS
- PD_VMR<*i*>, one for each VM<*i*>, where *i* is 0 to NUMVMBANK-1. PD_VMR<i> does not exist when NUMVMBANK = 0.

BR_CPU<*n*>

Each is controlled by a single PPU, the CPU<*n*>_PPU. The power domains in BR_CPU<*n*> are:

- PD_CPU<*n*>
- PD_CPU<*n*>EPU
- PD_CPU<*n*>RAM
- PD_CPU<*n*>TCM

BR_NPU<m>

Each NPU bounded region is controlled by a single PPU, the NPU<m>_PPU. The power domain in BR_NPU<m> is the PD_NPU<m>

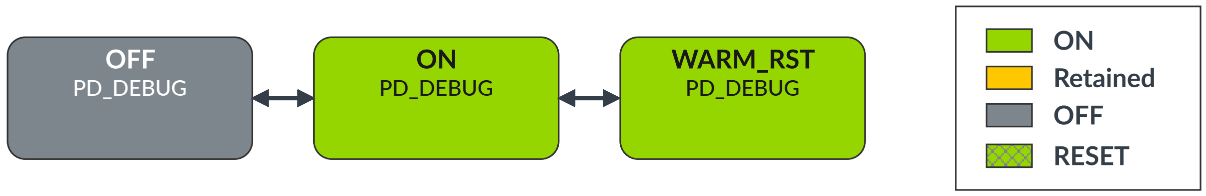
5.13.2.2 BR_DEBUG power modes

On Cold reset, the PD_DEBUG automatically transitions to ON state. If the debug system is not required to stay ON it eventually enters OFF state. It is woken either through the PD_DEBUG's Power Control Wakeup Q-Channel Device interface, or through a bus access targeting its shared debug domain.

The WARM_RST state is entered to bring the debug domain into an idle state so that the main system can be Warm reset cleanly. The WARM_RST state is entered, even though the debug domain itself is not being Warm reset at the same time.

The following figure shows the power modes that BR_DEBUG supports.

Figure 5-16: BR_DEBUG power mode transition diagram

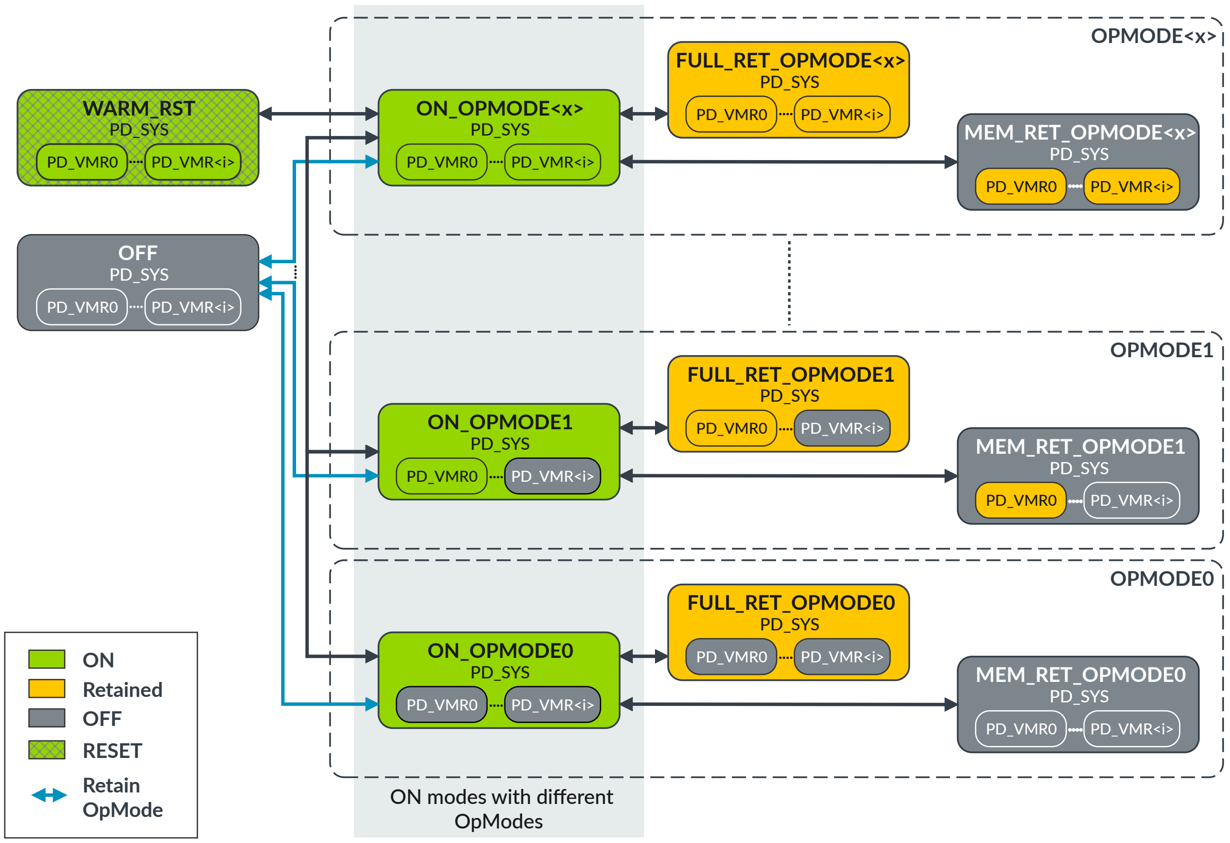


5.13.2.3 BR_SYS power modes

The BR_SYS supports several power modes.

The following figure shows the supported power modes:

Figure 5-17: BR_SYS power mode transition diagram



BR_SYS power modes have $2^{NUMVMBANK}$ different operating modes, which are 1, 2, 4, 8, or 16. This is encoded as a binary value of up to four bits, with each bit representing a power domain, the PD_VMR<i>. Each bit, *i*, indicates a specific PD_VMR<i> to be enabled or disabled.

When NUMVMBANK = 0, the BR_SYS power modes only have one operating mode. This means that a PPU supporting the BR_SYS does not need to support operating modes. When a specific PD_VMR<i> is disabled, its lower power mode is always in the OFF state. Other than the OFF or WARM_RST states, it is only possible to move between the operating modes while in one of the ON modes.

For example, for a system with NUMVMBANK = 2, the operating modes are:

OPMODE0

The PD_VMR0 and the PD_VMR1 are both disabled.

OPMODE1

The PD_VMR0 is enabled and the PD_VMR1 is disabled.

OPMODE2

The PD_VMR0 is disabled and the PD_VMR1 is enabled.

OPMODE3

The PD_VMR0 and the PD_VMR1 are both enabled.

On Cold reset, the system transitions to the ON_OPMODE<x> where *x* is $2^{NUMVMBANK}-1$. In this power mode, all the PD_VMR<i> are enabled if they exist.

If a Warm reset is requested, the bounded region transitions to the WARM_RST through other power modes when the domain is idle and ready for reset.

Note

The DMA includes a minimum power register in its programmers model. If the DMA is included in the system (NUMDMA > 0), it can prevent the PD_SYS from transitioning to a power state that is lower than its internal minimum power register allows.

5.13.2.3.1 Controlling PD_VMR<i> power mode

To configure the required low-power state of each PD_VMR<i>, the software must configure the register field PDCM_PD_VMR<i>_SENSE.MIN_PWR_STATE as follows:

`0b00` to set the low-power state of the PD_VMR<i> to OFF

The PD_VMR<i> only ever transitions between the ON and OFF states. It is never in retention. This means that in low-power state all state in the VM<i> is lost.

With this setting, when the PD_SYS is ON and the BR_SYS is in one of the ON_OPMODE modes that currently has the PD_VMR<i> ON, it transitions to another ON_OPMODE. In this ON_OPMODE the PD_VMR<i> is OFF automatically when the PD_VMR<i> domain is idle. Therefore, expect to see the PD_VMR<i> turn off quickly after the PDCM_PD_VMR<i>_SENSE.MIN_PWR_STATE is set to `0b00`.

When the PD_VMR<i> is OFF, it can only be returned to ON by an access on the bus targeting the PD_VMR<i>. Following that, when the PD_VMR<i> is idle again, it turns off again. To avoid this, configure the PDCM_PD_VMR<i>_SENSE.MIN_PWR_STATE to a nonzero value before accessing the VM<i>.

`0b01` to set the low-power state of the PD_VMR<i> to Retention

The PD_VMR<i> only ever transitions between the ON and Retention state, and is never OFF. This means that in low-power state, all register states in the VM<i> are retained. Therefore, for the BR_SYS, it never transitions to a power mode that has the PD_VMR<i> turned off. To place the PD_VMR<i> into Retention, the BR_SYS power modes have to enter either:

- One of the FULL_RET_OPMOE modes
- The MEM_RET_OPMODE mode where the PD_VMR<i> is in Retention

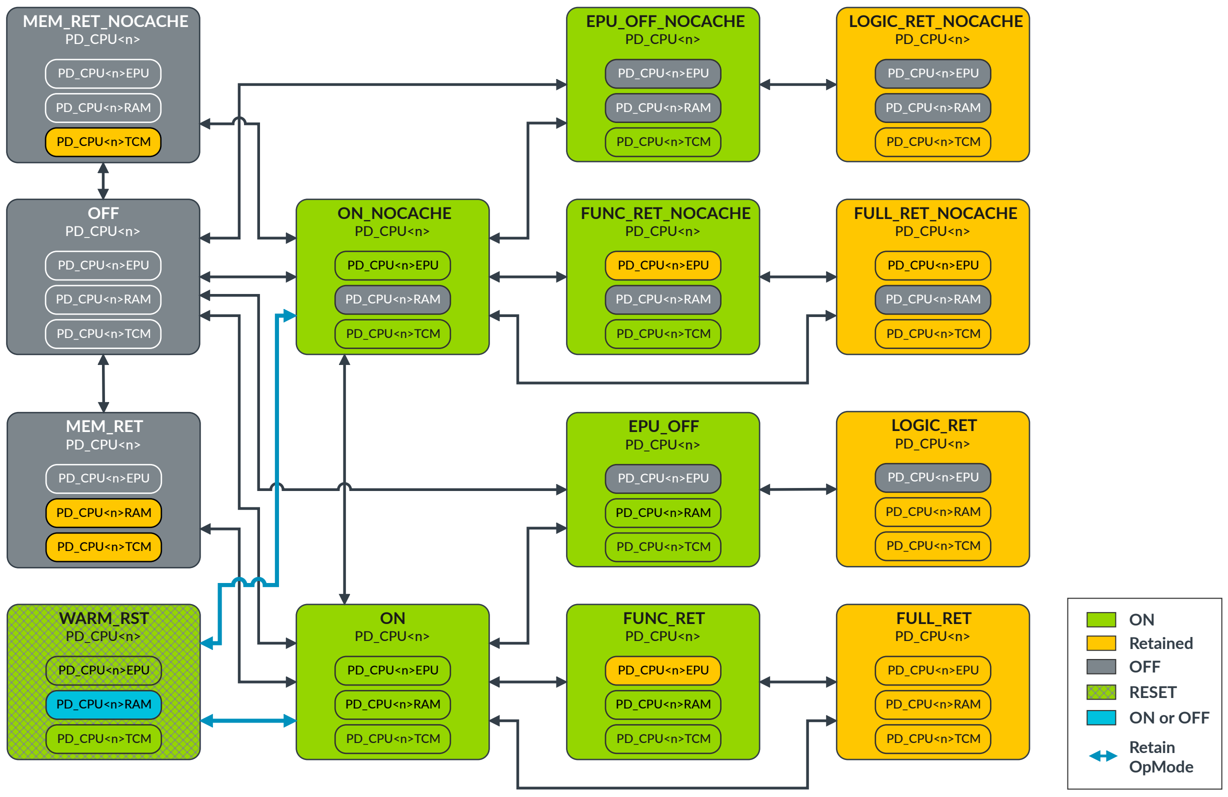
This means that it is not possible to place a PD_VMR into retention while keeping the PD_SYS ON.

5.13.2.4 BR_CPU<n> power modes

The BR_CPU<n> supports several power modes.

The following figure shows the supported power modes:

Figure 5-18: BR_CPU<n> power mode transition diagram



On Cold reset, the bounded region enters the EPU_OFF_NOCACHE power mode.

The PD_CPU<n>TCM power state always matches that of the PD_CPU<n>, except when in the MEM_RET or MEM_RET_NOCACHE states. In these states it is retained. The choice from the ON_NOCACHE to either the MEM_RET_NOCACHE or the OFF is determined by the local TCM minimum power state register configuration of the processor, the CPUPWRCFG.TCM_MIN_PWR_STATE.

If a Warm reset is requested, the bounded region transitions to the WARM_RST through other power modes when the domains are idle and ready for reset.

5.13.2.4.1 Controlling PD_CPU<n>RAM power states

The BR_CPU<n> bounded region uses operating modes to support the ability to turn on or off the cache RAMs in modes other than the OFF mode. Power modes with cache RAMs disabled, called the NOCACHE operating modes, are suffixed with NOCACHE. Power modes with cache RAMs enabled, called the CACHE operating modes, are without the NOCACHE suffix. To select the use of the NONCACHE operating modes, the following registers must be configured:

- Set CPDLPSTATE.RLPSTATE register in the processor to OFF which is `0b11`
- Set MSCR.DCACTIVE register in the processor to `0b0` to disable the Data Cache
- Set MSCR.ICACTIVE register in the processor to `0b0` to disable the Instruction Cache

Transitions between the two operating modes can only occur between the ON and ON_NOCACHE power modes.

When in the NOCACHE operating modes, the PD_CPU<n>RAM is always turned off. When operating in the CACHE operating modes, the PD_CPU<n>RAM enters RAM retention when entering the MEM_RET, LOGIC_RET, or FULL_RET power modes.

5.13.2.4.2 Controlling PD_CPU<n>EPU power states

Other than the WARM_RST state, the BR_CPU<n> bounded region provides the ability for the PD_CPU<n>EPU to enter a lower power state independently as long as the PD_CPU<n> is ON. To control if the PD_CPU<n>EPU remains ON, or be allowed to enter the Retention (RET) or OFF state, the software can use the register CPDLPSTATE.ELPSTATE in conjunction with the CPPWR.SU10 in the processor as follows:

RET

To allow the PD_CPU<n>EPU to enter the Retention state where the BR_CPU<n> never enters the EPU_OFF, LOGIC_RET EPU_OFF_NOCACHE, LOGIC_RET_NOCACHE, MEM_RET, MEM_RET_NOCACHE and OFF states, set either:

- CPDLPSTATE.ELPSTATE to RET, `0b10`
- CPDLPSTATE.ELPSTATE to OFF, `0b11`. Also set the CPPWR.SU10 to `0b0`. Retention is only entered when the CPU<n> is sleeping using WFI or WFE.

OFF

To allow the PD_CPU<n>EPU to enter the OFF state where the BR_CPU<n> never requests the FUNC_RET, FUNC_RET_NOCACHE, FULL_RET_NOCACHE, and FULL_RET states:

- Set the CPDLPSTATE.ELPSTATE to OFF, `0b11`. Also set the CPPWR.SU10 to `0b1`.

ON

To keep the PD_CPU<n>EPU in the ON, or context retained state where the BR_CPU<n> never requests the EPU_OFF, LOGIC_RET EPU_OFF_NOCACHE, LOGIC_RET_NOCACHE, MEM_RET, MEM_RET_NOCACHE, FUNC_RET, FUNC_RET_NOCACHE and OFF states:

- Set the CPDLPSTATE.ELPSTATE to ON, which is `0b00` or `0b01`.

Note

Depending on the processor core and the system implementation, other conditions may be required to reach the desired state.

For more information on these conditions, see the power control of the processors in, *Arm® Cortex®-M55 Processor Technical Reference Manual*, *Arm® Cortex®-M55 Processor Integration and Implementation Manual*, and *Arm® Cortex®-M85 Processor Integration and Implementation Manual*.

For more information on CPPWR, see *Arm®v8-M Architecture Reference Manual*.

5.13.2.4.3 Controlling the PD_CPU<n> power states

For the PD_CPU<n> to enter a lower power state, the software on the CPU<n> must first configure its CPDLPSTATE.CLPSTATE register. This register defines what power state the CPU<n> can enter when in a sleep state.

The possible power states that the CPU<n> can enter when in a sleep state are as follows:

Set the CPDLPSTATE.CLPSTATE to RET

To allow the PD_CPU<n> to enter retention state only when in sleep state. When set to RET, the BR_CPU<n> never enters the OFF, MEM_RET_NOCACHE, or MEM_RET modes. Other modes can be entered depending on the PD_CPU<n>EPU and the PD_CPU<n>RAM domain power states.

Set the CPDLPSTATE.CLPSTATE to OFF

To allow the PD_CPU<n> to enter the OFF state only when in sleep state. When set to OFF, the BR_CPU<n> never requests the LOGIC_RET_NOCACHE and the LOGIC_RET modes. Other modes can be entered if a coprocessor (CP<i>), which is included in the system, is indicating that the state cannot be lost (CPPWR.SU<i>=`0b0`). Entering other modes dependson on the PD_CPU<n>EPU and the PD_CPU<n>RAM domain power states.

Set the CPDLPSTATE.CLPSTATE to ON, that is, to `0b00` or `0b01`

To keep the PD_CPU<n> ON even when it is in sleep mode. This in turn means that the BR_CPU<n> never enters the LOGIC_RET_NOCACHE, LOGIC_RET, FULL_RET_NOCACHE, FULL_RET, OFF, MEM_RET_NOCACHE, and MEM_RET modes. Other modes can be entered depending on the PD_CPU<n>EPU and PD_CPU<n>RAM domain power states.

Note

Depending on the processor core and the system implementation, other conditions may be required to reach the desired state. For more information on these conditions, see the power control of the processors in *Arm® Cortex®-M55 Processor Technical Reference Manual*, *Arm® Cortex®-M55 Processor Integration and Implementation Manual*, and *Arm® Cortex®-M85 Processor Integration and Implementation Manual*.

For more information on CPPWR, see *Arm®v8-M Architecture Reference Manual*.

For each processor to then enter a lower power state, the processor must enter WFI DeepSleep state. In the CRSAS Ma2, DeepSleep always uses a WIC. External WIC for each processor always exists, and all external WICs reside in the PD_AON domain. Depending on the HASCPU<n>IWIC:

- If the HASCPU<n>IWIC is set to `0b0` for CPU<n>, then the internal WIC does not exist for the CPU<n>. In this case the external WIC is always used for CPU<n>.
- If the HASCPU<n>IWIC is set to `0b1` for the CPU<n>, then the internal WIC also exists. In this case the choice of which WIC to use is a software choice through the CPUPWRCFG.USEIWIC. For more information, CPUPWRCFG.

Any internal WIC that exist resides in the associated PD_CPU<n> power domain.

The types of power states each processor in the CRSAS Ma2 supports with information on how to enter each are as follows:

OFF - DeepSleep state

Allows the processor to turn off, but utilises interrupts through the external WIC to wake the processor. To enter this state, the processor must do the following before entering the WFI:

1. Select to use the external WIC through the CPUPWRCFG.USEIWIC register
2. Set the CPDLPSTATE.CLPSTATE of the CPU<n> to OFF
3. Enable DeepSleep

If the internal WIC is selected, the PD_CPU<n> is not able to turn off and remains ON.

RET - DeepSleep state

Allows the processor to enter the retention state, and utilises interrupts through the external WIC to wake the processor. To enter this state, the processor must do the following before entering WFI:

1. Select to use the external WIC through the CPUPWRCFG.USEIWIC register
2. Set the CPDLPSTATE.CLPSTATE to RET
3. Enable DeepSleep

If the internal WIC is selected instead, the PD_CPU<n> is not able to enter retention and remains ON.

ON - DeepSleep state

Allows the processor to enter a sleep state that only supports stopping the processor clock internally, including to the NVIC. It can use either the external or the internal WIC to wake the processor. Selection of external or internal WIC can be performed using the CPUPWRCFG.USEIWIC register prior to entering WFI.

To enter this state, the processor must do the following before entering WFI:

1. Set the CPDLPSTATE.CLPSTATE to `0b01`, ON
2. Enable DeepSleep

ON - Sleep state

Allows the processor to enter a sleep state that is still ON, keeping its NVIC clocking and running with the rest of the core clock turned off. To enter this state, before entering WFI or WFE the processor:

- Must not enable DeepSleep
- Must not set the CPDLPSTATE.CLPSTATE to ON, `0b00`

ON state

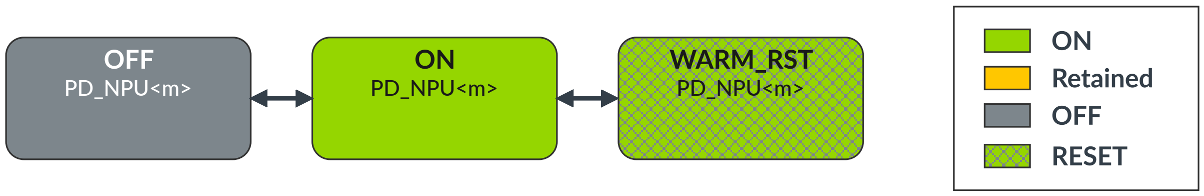
The normal running state of the processor. In this state, the EPU, and the RAMs in the processor have a degree of separate control as detailed in Controlling PD_CPU<n>RAM power states and Controlling PD_CPU<n>EPU power states.

5.13.2.5 BR_NPU<m> power modes

The BR_NPU<m> supports several power modes.

The following figure shows supported power modes:

Figure 5-19: BR_NPU power mode transition diagram



On Cold reset, the bounded region enters the ON state.

If a Warm reset is requested, the bounded region transitions to the WARM_RST when all dependent domain is idle and ready for reset. The CMD register of the NPU can be programmed to prevent the NPU powering off by setting its power_q_enable field to 0.

For more information of this register, see *Arm® Ethos™-U55 NPU Technical Reference Manual* or *Arm® Ethos™-U65 NPU Technical Reference Manual*.

After reset, the NPU does not block power down, after it enters an idle state.

5.13.2.6 Intermediate power dependency control

The power dependency control implements a Power Dependency Control Matrix (DPCM) which is used to describe and control how each power domain affects another and the lowest power state each domain is allowed to enter. It allows, as much as possible, for the power control of the system to be performed primarily using dynamic power transitions. This reduces the number of software interactions needed for system management. This improves its responsiveness and contributes to further power reduction. The PDCM and all other sensitivity are defined for each power domain.

Table 5-9: Power Dependency Matrix for PILEVEL = 1 on page 99 shows the PDCM table how the PD_SYS and PD_VMR<*i*> are affected by the power state of the other domains.

The heading row of the table lists the Power Domains that are being controlled, while the left column lists the Power dependency inputs. Power dependency inputs are either:

- The “ON” state of power domains in the system. For example, the PD_SYS_ON means that the PD_SYS is ON when asserted.
- Expansion Power Dependency Control Matrix Q-Channel signals. The PDCMONQREQn signal and the PDCMRETQREQn signals are driven by expansion logic from outside the subsystem. This indicates a keep-up request from external power domains.

“Conf”

Indicates that the power domain is software configurable

“Y”

Indicates that the power domain is always sensitive to the respective dependency input.

For example, the PD_SYS can be software configured to be sensitive to the ON state of the PD_SYS and all Expansion Power Control Dependency inputs. However it is always sensitive to the ON state of all the PD_CPU<*n*> and PD_NPU<m>.

When a power domain is sensitive to an ON dependency input, it means the following:

- When the controlled power domain is already ON and any of the dependency inputs is ON or true, then the power domain remains ON. The exception is with the PDCMRETQREQn inputs. With these inputs, if a power domain is configured to be sensitive, the power domain maintains at least in a retention state. The PDCM is used primarily to define when a power domain should not enter a lower power state. It is not designed to support powering up of any power domain.

Note

In Table 5-9: Power Dependency Matrix for PILEVEL = 1 on page 99, the following apply:

- PD_VMR<i> columns do not exist if *i* > (NUMVMBANK – 1)
- PD_CPU<*n*>_ON row does not exist if *n* > NUMCPU
- PD_NPU<*m*>_ON row does not exist if NUMNPU = 0
- In the cases of PDCMONQREQn[k] and PDCMRETQREQn[k], where k is {0-<PDCMQCHWIDTH-1>}

Table 5-9: Power Dependency Matrix for PILEVEL = 1

Power Dependency Inputs/ Power Domain	PD_SYS	PD_VMR0[1]	PD_VMR1[1]	PD_VMR2[1]	PD_VMR3[1]
PD_SYS_ON	Conf	-	-	-	-
PD_CPU0_ON	Y	Conf	Conf	Conf	Conf
PD_CPU1_ON	Y	Conf	Conf	Conf	Conf

Power Dependency Inputs/ Power Domain	PD_SYS	PD_VMR0[1]	PD_VMR1[1]	PD_VMR2[1]	PD_VMR3[1]
PD_CPU2_ON	Y	Conf	Conf	Conf	Conf
PD_CPU3_ON	Y	Conf	Conf	Conf	Conf
PD_NPU0_ON	Y	Conf	Conf	Conf	Conf
PD_NPU1_ON	Y	Conf	Conf	Conf	Conf
PD_NPU2_ON	Y	Conf	Conf	Conf	Conf
PD_NPU3_ON	Y	Conf	Conf	Conf	Conf
PDCMONQREQn[k]	Conf	Conf	Conf	Conf	Conf
PDCMRETQREQn[k]	Conf	Conf	Conf	Conf	Conf

The PD_VMR<*i*> can also be configured to be sensitive to a power domain. For example the memory also remains ON, if both are true:

- The PD_VMR0 sensitivity is configured by the software to be sensitive to the PD_CPU0_ON
- The PD_CPU0 is ON

However, the PD_VMR0 is controlled using the same PPU as the PD_SYS. As a result of the power state of the PPU shown in the figure in BR_SYS power modes, whenever the PD_VMR0 is ON, the PD_SYS also remains in one of the ON_OPMODE states where the PD_VMR0 is ON.

Note

The PD_SYS can be configured to be sensitive to itself. When the software configures a domain to be sensitive to itself, the domain remains ON after it is ON.

The CRSAS Ma2 also provides a programmable register for the power domains shown in the following table, which defines the lowest power mode that each domain can enter. The PD_NPU<m> and the PD_DEBUG defaults to a minimum power state of OFF.

Table 5-10: Power Domain's Minimum Power State for PILEVEL = 1

Power Domain	Supported MIN_PWR_STATE for each domain
PD_SYS	ON, OFF, Retention
PD_VMR<*i*>	ON, OFF, Retention

The MIN_PWR_STATEs of both the PD_SYS and the PD_VMR<*i*> help to determine which power modes of the BR_SYS PPU can be used next when performing dynamic power transition.

For example, if all the following are true:

- The system is idle
- All dependent domains are not ON
- Then the BR_SYS PPU tries to enter the bounded region collectively to a low-power state
- If MIN_PWR_STATE of the PD_SYS is set to Retention

Then the BR_SYS is never allowed to enter the OFF state nor any of the theMEM_RET_OPMODE<x> states. This is because the PS_SYS is not allowed to turn off. In tis case, depending on the current PD_VMR<*i*> states, it tries to enter one of the associated FULL_RET_OPMODE<x> states.

In another example, if all the following are true:

- PDCM_PD_VMR<x>_SENSE.MIN_PWR_STATE of the PD_VMR0 is "ON"
- All other PD_VMRs are OFF
- If currently all PD_VMR<*i*> are ON when the BR_SYS PPU is entering a low-power state

Then it transitions to ON_OPMODE1 state to turn off all the PD_VMR<*i*>, except for the PD_VMR0. It never enters the FULL_RET_OPMODE1 nor the MEM_RET_OPMODE1.

5.13.2.7 Intermediate System Level Power States

Through the relationship defined by the power dependency control matrix and the processor minimum power states, we can define several System Power States that the system supports.

The supported System Power States are as follows:

SYS_OFF

All voltage and power domains are OFF.

HIBERNATION0

The VSYS voltage domain is ON. The system is in the lowest power state that can still be woken from sleep. At wake, the system has to reboot.

SYS_RET

The system is in retention. At wake, the system can continue to execute since no system state is lost.

SYS_ON

The system is ON.

Table 5-11: System Level Power States when PILEVEL = 1 on page 102 lists the power states of each power domain that each of the System Power State supports or requires. See Controlling PD_CPU<n> power states.

Note

In Table 5-11: System Level Power States when PILEVEL = 1 on page 102, the following apply:

- When the CPU<n> and therefore the PD_SYS is in OFF-DeepSleep state, the PD_CPU<n>TCM can either be in:
    - OFF
    - Retention and is defined by CPUPWRCFG.TCM_MIN_PWR_STATE. In this state, the EPU must be OFF. The PD_CPU<n>RAM can be OFF permanently and is defined by CPDLPSTATE.RLPSTATE.
- The PD_NPU<m> power domain only exists when NUMNPU > 0.

- The PD_VMR<i> power domain only exists if NUMVMBANK > 0.

Table 5-11: System Level Power States when PILEVEL = 1

Power Domains	SYS_OFF	HIBERNATION 0	SYS_RET	SYS_ON
VSYS (PD_AON)	OFF	ON	ON	ON
PD_SYS	OFF	OFF	RET	ON
PD_CPU<n>	OFF	OFF - DeepSleep	OFF - DeepSleep RET - DeepSleep	OFF - DeepSleep RET - DeepSleep ON - DeepSleep ON - Sleep ON
PD_NPU<m>	OFF	OFF	OFF	OFF/ON
PD_DEBUG	OFF	OFF/ON	OFF/ON	OFF/ON
PD_VMR<i>	OFF	OFF/RET	OFF/RET	OFF/ON

The following points can be made based on the Table 5-11: System Level Power States when PILEVEL = 1 on page 102:

- The System Level Power States are closely associated with the PD_SYS power states (PD_CPU<n>EPU and PD_CPU<n>RAM), and the PD_VMR<i> supported states are associated with the PD_SYS states. See the figure in BR_SYS power modes.
- Generally, the PD_DEBUG can be woken independently, except in the SYS_OFF state.
- The memory power states (PD_VMR) are defined as part of the BR_SYS and cannot be controlled independently from the PD_SYS.

An additional transient WARM_RST state exists for the system when a Warm reset is being performed. This state can only be entered from the SYS_ON state. In this state, all power domains, except for the PD_AON, temporarily enter WARM_RST mode and exit it when Warm reset is completed.

5.13.3 Basic level power infrastructure

The basic level power infrastructure is when PILEVEL = 0. This locks the processor top-level power domain to the main system to reduce the complexity of the infrastructure. This reduces the number of power domains and hence the number of PPUs in the system.

Therefore, this simlification removes the ability to separately control the power state of the processor, the system, and volatile memories in the system. In addition, such a system can only support a single processor. Therefore the NUMCPU must be '0' when PILEVEL = 0.

5.13.3.1 Power hierarchy

The system at PILEVEL = 0 supports a single voltage domain, and several hierarchical power domains.

The system supports the following voltage domain:

VSYS

The system voltage domain.

The system supports the followingl power domains:

- PA_AON
- PD_SYS
- PD_NPU<*m*>, does not exist if NUMNPU = 0
- PD_VMR<*i*>, where i is 0 to NUMVMBANK-1, and does not exist if NUMVMBANK = 0
- PD_CPU0EPU
- PD_CPU0RAM

The following figure shows the voltage and power domain hierarchy:

Figure 5-20: Voltage and Power domain hierarchy of a subsystem with PILEVEL = 0

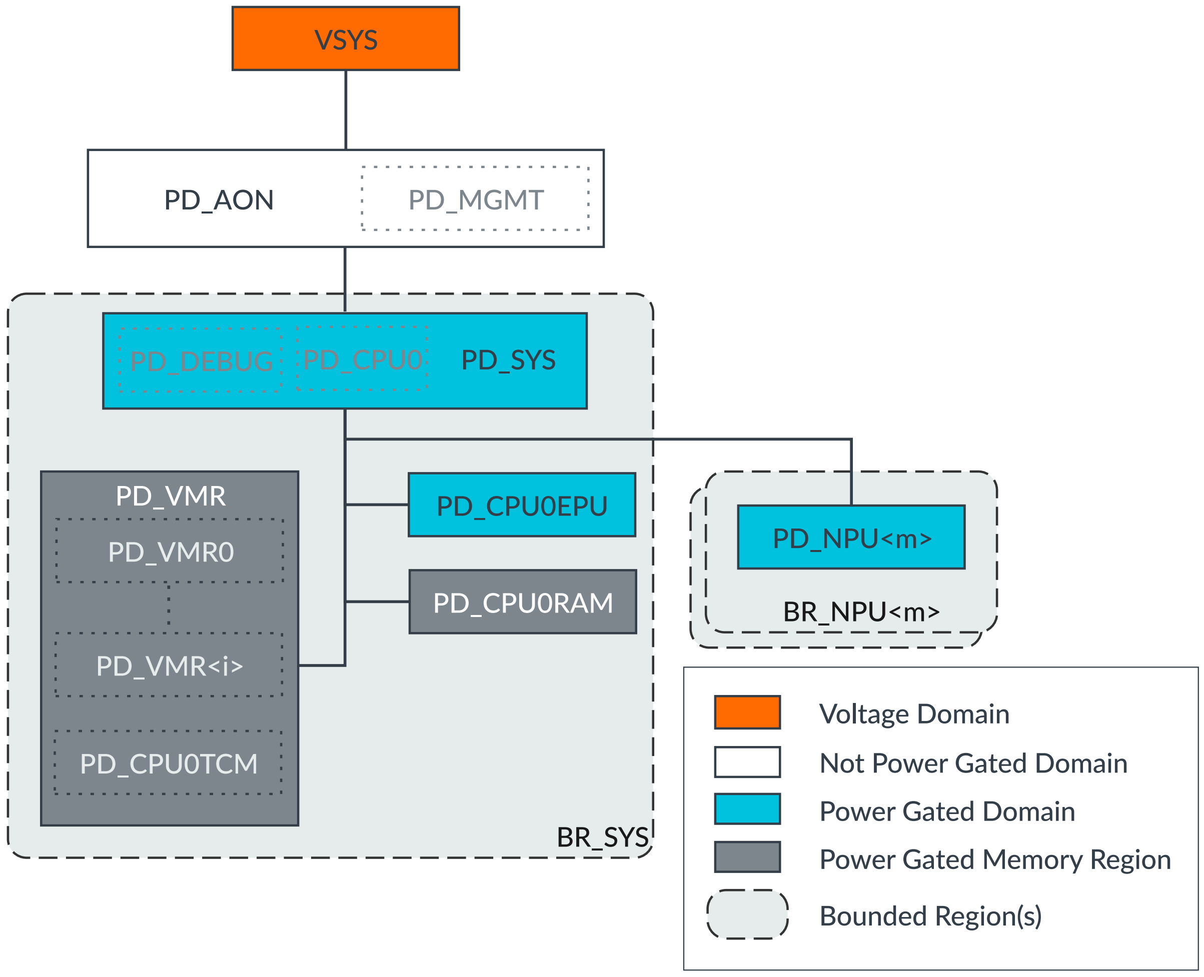


Figure 5-20: Voltage and Power domain hierarchy of a subsystem with PILEVEL = 0 on page 104 shows several bounded regions. Each of these bounded regions are controlled by a single PPU. The bounded regions and their controlling PPU are as follows:

BR_NPU<m>

This is controlled by the single PPU, NPU<m>_PPU. The power domain in BR_NPU<m> is the PD_NPU<m>

BR_SYS

This is controlled by the single PPU, SYS_PPU. The power domains in BR_SYS are:

- PD_SYS. Note that what was PD_DEBUG is now merged into PD_SYS.
- PD_CPU0EPU
- PD_CPU0RAM
- PD_VMR which contains all the volatile memory instances

The PD_CPU in PILEVEL = 1 is here merged with the PD_SYS. The PD_SYS is controlled by the PPU that controls the PD_CPU power domain. The PD_CPU0TCM power domain is merged with all the PD_VMR<*i*> where *i* is 0 to NUMVMBANK – 1, to form a single PD_VMR domain. This PD_VMR domain is controlled using the same PPU as the TCM memory in PILEVEL = 1. Because of this, at minimum configuration all the power control can be implemented with only a single PPU. This reduces complexity and area at the cost to flexibility.

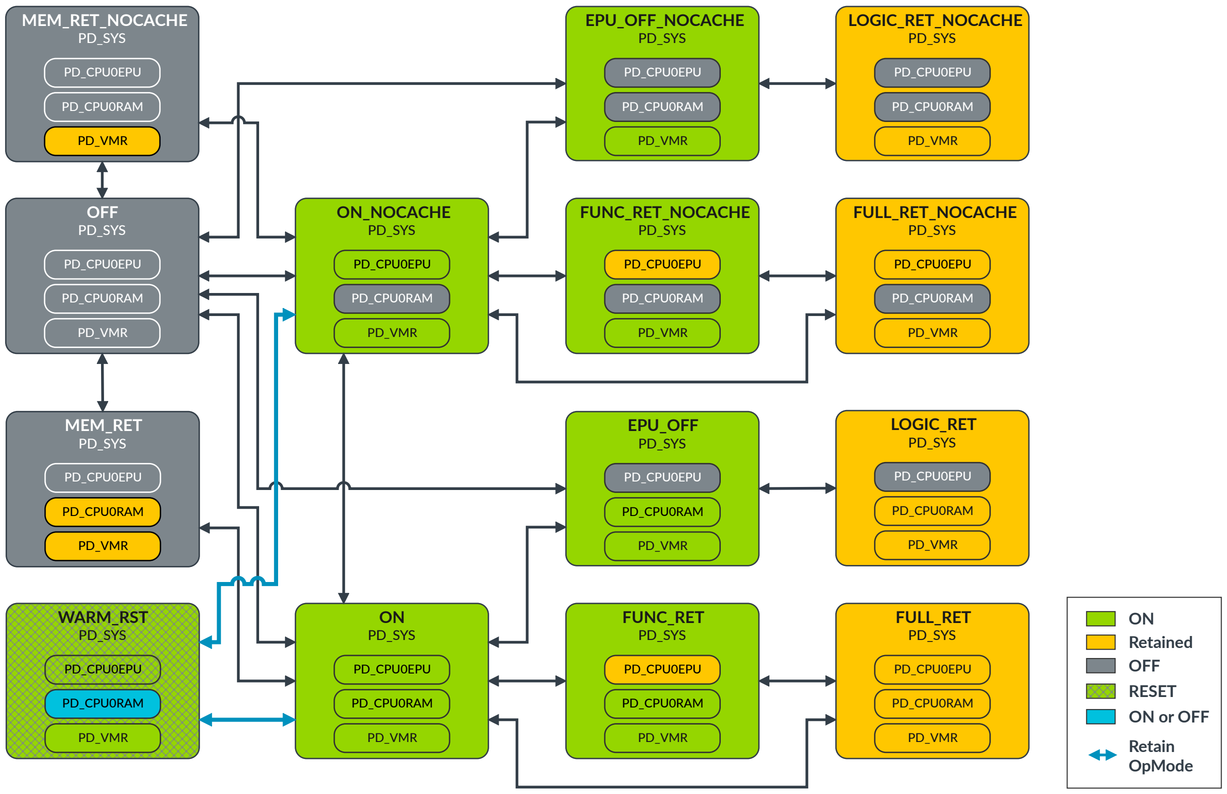
5.13.3.2 BR_SYS power modes

The BR_SYS supports several power modes. Because the PD_SYS is merged with the PD_CPU0 when PILEVEL = 0, the BR_SYS has a power mode transition diagram like the BR_CPU<n> when PILEVEL = 1.

In addition, the PD_SYS is also merged with the PD_DEBUG. Therefore, if the DEBUGPWR Q-Channel or P-Channel Expansion Device interface exist, it will behave in the same way as the SYSPWR Q-Channel or P-Channel Expansion Device interface. In the same way, if PWRDEBUGWAKE Q-Channel interface exist, it will behave in the same way as the PWRSYSWAKE Q-Channel interface. This allows debug expansion logic to be added to the PD_SYS and for other expansion logic to wake the PD_SYS.

The following figure shows the supported power modes:

Figure 5-21: BR_SYS power mode transition diagram



On Cold reset, the bounded region enters the EPU_OFF_NOCACHE power mode.

The PD_VMR power state always matches that of the PD_SYS, except when in MEM_RET or MEM_RET_NOCACHE state where it is retained. The choice from the ON_NOCACHE to either MEM_RET_NOCACHE or OFF is determined by the CPU 0 local TCM minimum power state register configuration (CPUPWRCFG.TCM_MIN_PWR_STATE). Therefore, at PILEVEL = 0, Volatile memory bank are treated like TCMs for power control.

If a Warm reset is requested, the bounded region transitions to the WARM_RST through other power modes when the domains are idle and ready for reset. The DMA includes a minimum power register in its programmers' model. If the DMA is included in the system (NUMDMA > 0) then it can prevent the PD_SYS from transitioning to a power state that is lower than its internal minimum power register allows.

5.13.3.2.1 Controlling the PD_CPU0RAM power states

The BR_SYS bounded region uses operating modes to support the ability to turn on or off the cache RAMs in modes other than the OFF mode.

- Power modes with cache RAMs disabled, called the NOCACHE operating modes, are suffixed with NOCACHE.
- Power modes with cache RAMs enabled, called the CACHE operating modes, are the modes without the NOCACHE suffix.

To select the use of the NONCACHE operating modes, the following registers must be configured:

- Set CPDLPSTATE.RLPSTATE register in the CPU to OFF which is `0b11`,
- Set MSCR.DCACTIVE register in the CPU to `0b0` to disable Data Cache,
- Set MSCR.ICACTIVE register in the CPU set to `0b0` to disable Instruction Cache.

Transitions between the two operating modes can only occur between the ON and ON_NOCACHE power modes.

When in the NOCACHE operating modes, the PD_CPU0RAM is always turned off. When operating in the CACHE operating modes, the PD_CPU0RAM enters RAM retention when entering the MEM_RET, LOGIC_RET, or FULL_RET power modes.

5.13.3.2.2 Controlling the PD_CPU0EPU power states

Other than the WARM_RST state, the BR_SYS bounded region provides the ability for the PD_CPU0EPU to enter a lower power state independently as long as the PD_SYS is ON. To control if the PD_CPU0EPU can remain ON or be allowed to enter the Retention (RET) or OFF state, software can use the register CPDLPSTATE.ELPSTATE in conjunction with the CPPWR.SU10 in the processor as follows:

RET

To allow the PD_CPU0EPU to enter the Retention state where the BR_SYS never enters the EPU_OFF, LOGIC_RET EPU_OFF_NOCACHE, LOGIC_RET_NOCACHE, MEM_RET, MEM_RET_NOCACHE and OFF states, set either:

- CPDLPSTATE.ELPSTATE to RET, `0b10`
- CPDLPSTATE.ELPSTATE to OFF, `0b11` and the CPPWR.SU10 to `0b0`.

Retention is only entered when CPU<n> is sleeping using WFI or WFE.

OFF

To allow the PD_CPU0EPU to enter OFF state where the BR_SYS never requests the FUNC_RET, FUNC_RET_NOCACHE, FULL_RET_NOCACHE, and FULL_RET states, set:

- CPDLPSTATE.ELPSTATE to OFF, `0b11`, and the CPPWR.SU10 to `0b1`.

ON

To keep the PD_CPU0EPU in ON or context retained state where the BR_SYS never requests the EPU_OFF, LOGIC_RET EPU_OFF_NOCACHE, LOGIC_RET_NOCACHE, MEM_RET, MEM_RET_NOCACHE, FUNC_RET, FUNC_RET_NOCACHE and OFF states, set:

- CPDLPSTATE.ELPSTATE to ON, which is `0b00` or `0b01`.

Note

Depending on the processor core and the system implementation, other conditions may be required to reach the desired state.

For more information on these conditions see the power control of the processors in, *Arm® Cortex®-M55 Processor Technical Reference Manual*, *Arm® Cortex®-M55 Processor Integration and Implementation Manual*, and *Arm® Cortex®-M85 Processor Integration and Implementation Manual*.

For more information on CPPWR, see *Arm®v8-M Architecture Reference Manual*.

5.13.3.2.3 Controlling PD_SYS Power States

For the system to enter a lower power state, the processor must enter the WFI DeepSleep state. In the CRSAS Ma2, DeepSleep always uses a WIC. EWIC of the processor always exists and resides in the the PD_AON domain. Depending on the HASCPU0IWIC:

- If set to `0b0`, then the internal WIC does not exist and the EWIC is always used for the CPU<*n*>.
- If set to `0b1`, then the internal WIC also exists, and the choice of which WIC to use is a software choice through the CPUPWRCFG.USEIWIC. See CPUPWRCFG.

Any internal WIC that exists resides in the PD_SYS power domain.

The types of power states each processor in the CRSAS Ma2 supports with information on how to enter each are as follows:

OFF - DeepSleep state

Allows the PD_SYS to turn off but utilise interrupts through the external WIC to wake the system. To enter this state, the processor must do the following before entering WFI:

- Select to use the external WIC through the CPUPWRCFG.USEIWIC register
- Set the CPDLPSTATE.CLPSTATE of the CPU0 to OFF
- Enable DeepSleep

Note that if Internal WIC is selected PD_CPU<n> is not able to turn off and remains ON.

RET - DeepSleep state

Allows the PD_SYS to enter the retention state and utilise interrupts through the external WIC to wake the system. To enter this state, the processor must do the following before entering WFI:

- Select to use the external WIC through the CPUPWRCFG.USEIWIC register
- Set the CPDLPSTATE.CLPSTATE to RET
- Enable DeepSleep

Note that if Internal WIC is selected instead, the PD_SYS is not able to enter retention and remains ON.

ON - DeepSleep state

Allows the PD_SYS to stay ON with the processor in a sleep that is only stopping the processor clock internally, including to the NVIC. It can use either the external or internal WIC to wake the processor. Selection of external or internal WIC can be performed using the CPUPWRCFG.USEIWIC register prior to entering WFI.

To enter this state, the processor must do the following before entering WFI:

- Set CPDLPSTATE.CLPSTATE to `0b01`, ON
- Enable DeepSleep before entering WFI.

ON - Sleep state

Allows the PD_SYS to stay ON with the processor in a sleep state that keeps its NVIC clocking and running, but with the rest of the processor core clock turned off. To enter this state, before entering WFI or WFE the processor:

- Must not enable DeepSleep
- Must not set the CPDLPSTATE.CLPSTATE to ON, `0b00`

ON state

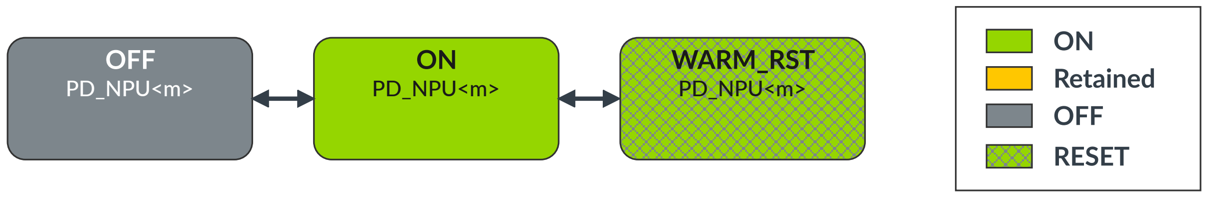
The normal running state of the PD_SYS and the processor. In this state the EPU and the RAMs in the processor have a degree of separate control as detailed in the previous sections.

5.13.3.3 BR_NPU<m> power modes

The BR_NPU<m> supports several power modes.

The following figure shows supported power modes:

Figure 5-22: BR_NPU power mode transition diagram



On Cold reset, the bounded region enters the ON state.

When a Warm reset is requested, the bounded region transitions to the WARM_RST when all dependent domain is idle and ready for reset. The CMD register of the NPU can be programmed to prevent the NPU powering off by setting its power_q_enable field to 0.

For more information of this register, see *Arm® Ethos™-U55 NPU Technical Reference Manual* or *Arm® Ethos™-U65 NPU Technical Reference Manual*.

When reset, the NPU does not block power down after it enters an idle state.

5.13.3.4 Basic power dependency control

The power dependency control implements a Power Dependency Control Matrix (DPCM) which is used to describe and control how each power domain affects another and the lowest power state each domain is allowed to enter. It allows, as much as possible, for the power control of the system to be performed primarily using dynamic power transitions. This reduces the number of software interactions needed for system management. This improves its responsiveness and contributes to further power reduction. The PDCM and all other sensitivity are defined for each power domain.

Table 5-12: Power Dependency Matrix, for PILEVEL = 0 on page 110 shows the PDCM table and how the PD_SYS is affected by the power state of the other domains.

The table only includes dependency controls for a single power domain, the PD_SYS. The left column lists the Power dependency inputs. Power dependency inputs are either:

- The “ON” state of power domains in the system. For example, the PD_SYS_ON means that the PD_SYS is ON when asserted.

- Expansion Power Dependency Control Matrix Q-Channel signals. The PDCMONQREQn signal and the PDCMRETQREQn signal are driven by expansion logic from outside the subsystem. This indicates a keep-up request from external power domains.

"Conf"

Indicates that it is software configurable to be sensitive to the respective dependency input.

For example, the PD_SYS can be software configured to be sensitive to the ON state of itself and all Expansion Power Control Dependency inputs.

When a power domain is sensitive to an ON dependency input, it means the following:

- When the controlled power domain is already ON and any of the dependency inputs is ON or true, then the power domain remains ON. The exception are the PDCMRETQREQn inputs. With these inputs, if a power domain is configured to be sensitive to these, the power domain maintains at least in a retention state. The PDCM is used primarily to define when a power domain should not enter a lower power state. It is not designed to support powering up of any power domain.

Note

In Table 5-12: Power Dependency Matrix, for PILEVEL = 0 on page 110, the following apply:

- PD_NPU<m>_ON row does not exist if NUMNPU = 0
- In the cases of PDCMONQREQn[k] and PDCMRETQREQn[k], [k] is {0-<PDCMQCHWIDTH-1>}

Table 5-12: Power Dependency Matrix, for PILEVEL = 0

Power Dependency Inputs/ Power Domain	PD_SYS
PD_SYS_ON	Conf
PD_NPU<m>_ON	Y
PDCMONQREQn[k]	Conf
PDCMRETQREQn[k]	Conf

Because the PD_SYS can be configured to be sensitive to itself, when set up by the software to do so, the PD_SYS remains ON after it is ON.

At PILEVEL = 0, the desired power state of the system is defined in part of the processor internal controls and states, and the choice between IWIC or EWIC. See Controlling PD_SYS power states.

5.13.3.5 Basic System Level Power States

Through the relationship defined by the power dependency control matrix, and the processor minimum power states, we can define several System Power States that the system supports, and these are:

SYS_OFF

All voltage and power domains are OFF.

HIBERNATION0

The VSYS voltage domain is ON, and the system is in the lowest power state that can still be woken from sleep. At wake, the system has to reboot.

SYS_RET

The system is in retention. At wake, the system can continue to execute since no system state is lost.

SYS_ON

The system is ON.

Table 5-13: System Level Power States when PILEVEL = 0 on page 111 the power states of each power domain that each of the System Power State supports or requires.

Note

In Table 5-13: System Level Power States when PILEVEL = 0 on page 111, the following apply:

- When the processor and therefore the PD_SYS is in OFF-DeepSleep state, the PD_VMR can either be in:
    - OFF
    - Retention and is defined by the CPUPWRCFG.TCM_MIN_PWR_STATE. In this state, the EPU must be OFF. The PD_CPU0RAM can be OFF permanently and is defined by the CPDLPSTATE.RLPSTATE.
- The PD_NPU<m> power domain only exist when NUMNPU > 0.

Table 5-13: System Level Power States when PILEVEL = 0

Power Domains	SYS_OFF	HIBERNATION 0	SYS_RET	SYS_ON
VSYS (PD_AON)	OFF	ON	ON	ON
PD_SYS	OFF	OFF - DeepSleep	RET - DeepSleep	ON - DeepSleep ON - Sleep ON
PD_NPU<m>	OFF	OFF	OFF	OFF/ON

The following points can be made based on the Table 5-13: System Level Power States when PILEVEL = 0 on page 111:

- The System Level Power States are closely associated with the PD_SYS power states (PD_CPU0EPU and PD_CPU0RAM), and the PD_VMR supported states are associated with the PD_SYS states. See BR_SYS power modes.
- Memory power states, PD_VMR are defined as part of BR_SYS and cannot be controlled independently from PD_SYS.

An additional transient WARM_RST state also exists for the system when a Warm reset is being performed. This state can only be entered from SYS_ON state. In this state, all power domains, except PD_AON temporarily enter WARM_RST mode and exit it when Warm reset is completed.

6. Interfaces

The subsystem has several interfaces. Some interfaces are only required under certain configurations.

It is **IMPLEMENTATION DEFINED** whether an interface which is not required is either:

- Not implemented
- Implemented but the signals are tied off

In this section, the following conventions are used:

AMBA manager interface

An AMBA manager interface is one where the subsystem is the manager and must be connected to a subordinate interface.

For an AXI manager interface the ARVALID signal is an output and the ARREADY signal is an input.

For an AHB manager interface the HADDR signal is an output and the HRDATA signal is an input.

For an APB manager interface, the PADDR signal is an output.

AMBA subordinate interface

An AMBA subordinate interface is one where the subsystem is the subordinate and must be connected to a manager interface.

For an AXI subordinate interface the ARVALID signal is an input and the ARREADY signal is an output.

For an AHB subordinate interface the HADDR signal is an input and the HRDATA signal is an output.

For an APB subordinate interface, the PADDR signal is an input.

For more information on AMBA protocols, see *AMBA® AXI and ACE Protocol Specification*, *Arm® AMBA® 5 AHB Protocol Specification*, and *Arm® AMBA® APB Protocol Specification*.

LPI control interface

An LPI control interface is one where the subsystem is the device and must be connected to a control interface.

For a Q-Channel control interface the QREQn signal is an input and the QACCEPTn, QDENY, and QACTIVE signals are outputs.

For a P-Channel control interface the PREQn signal is an input and the PACCEPTn, PDENY, and PACTIVE signals are outputs.

LPI device interface

An LPI device interface is one where the subsystem is the controller and must be connected to a device interface.

For a Q-Channel device interface the QREQn signal is an output and the QACCEPTn, QDENY and QACTIVE signals are inputs.

For a P-Channel device interface the PREQn signal is an output and the PACCEPTn, PDENY and PACTIVE signals are inputs.

An implementation of the CRSAS Ma2 can add other interfaces as long as it does not affect the behaviour of the AMBA and LPI interfaces. For example, the processor core used in this system often provides more processor state outputs. These can always be made available for expansion but their existence is **IMPLEMENTATION DEFINED**.

6.1 Input and output clocks

The subsystem has several input and output clocks.

The following table lists all the clock inputs into the subsystem.

Table 6-1: The CRSAS Ma2 input clocks

Clock Name	Target Power Domain PILEVEL= 2	Description
SLOWCLK	PD_AON	Slow Clock. An always active slow clock input that is completely asynchronous to the other clocks in the system. This is one of the only two clocks expected to be active in the lowest power state of the system, HIBERNATE{0,1}, and is used primarily by timers that reside in the PD_AON domain.
AONCLK	PD_AON	Always ON Clock Input. This clock is used for logic in the PD_AON domain that is not running on SLOWCLK. This allows the rest of the PD_AON domain to run on a faster clock and yet be independent from the rest of the system.
CNTCLK	PD_AON	System Counter Timestamp Clock associated with the CNTVALUE<G/B> System Counter Timestamp input.
SYSCLK	PD_MGMT	Main System Clock Input. This clock is the main clock used to drive the main system that resides in the PD_SYS. This clock is also used for logic in the PD_MGMT domain if it exists, or logic that is merged from the PD_MGMT to the PD_AON when PILEVEL < 2. When: **PILEVEL = 1** PD_MGMT is merged with the PD_AON and all reference to the PD_MGMT is replaced by the PD_AON **PILEVEL = 0** PD_MGMT is merged with the PD_AON and all reference to the PD_MGMT is replaced by the PD_AON
DEBUGCLK	PD_DEBUG	Debug System Clock Input. This clock is used to drive all logic in the debug System. This input clock must exist when HASCSS = 1.

Clock Name	Target Power Domain PILEVEL= 2	Description
CPU<n>CLK	PD_CPU<n>	Processor clock. This is the clock used to drive each processor. When: **PILEVEL = 1** PD_CPU<n> is merged with the PD_SYS and all reference to the PD_CPU<n> is replaced by the PD_SYS
NPU<m>CLK	PD_NPU<m>	NPU clock. This is the clock used to drive each NPUs. This clock only exists if NUMNPU > 0.

The relation between these clocks is **IMPLEMENTATION DEFINED** with the architecture able to support all clocks being completely asynchronous to each other. During implementation the implementer or the SoC integrator can drive several clocks using the same clock sources. This can be done to reduce the number of clock sources. If the implementer or the SOC integrator chooses to do so, they must take clock availability, power, and response time into consideration.

In typical use, we recommend that the SLOWCLK is driven using a 32kHz clock source. If reducing standby power is an important consideration for a product, the AONCLK can be driven at a lower clock rate (around 1MHz to 10MHz) compared to SYSCLK. This improves the transition time entering and leaving the lowest power state of the system, but still supports the use of very low leakage implementation library cells.

Alternatively, the AONCLK can be driven using the same clock source as the SYSCLK. If there is no requirement to run the processors and System Timestamp Counter at a different speed to the system, then the CPU<n>CLK and the CNTCLK can also be driven using the same clock source as the SYSCLK. The DEBUGCLK can also be driven using the same clock as the CPU<n>CLK. The NPU<m>CLK can be driven from the same clock source as the SYSCLK.

At a minimum, two clock sources are needed, for example, one at 32KHz for the SLOWCLK and another at 200MHz for the other clocks. The SYSCLK can be clocked synchronously to and at a fixed fraction of the CPU<n>CLK, in cases like when large SRAMs with higher access time are required.

Note

All clocks always enter through the PD_AON domain first before being used in their respective power domain.

The following table lists all the clock outputs from the subsystem, and the target power domain that each clock is expected to be used for.

Table 6-2: The CRSAS Ma2 output clocks

Clock Name	Power Domain expected to be used in PILEVEL = 2	Description
MGMTSYSCLK	PD_MGMT	Gated version of the SYSCLK expected to be used to drive expansion logic that resides in the the PD_MGMT power domain, when PILEVEL = 2. When: **PILEVEL = 0** PD_MGMT is merged with the PD_AON and all reference to the PD_MGMT is replaced by the PD_AON **PILEVEL = 1** PD_MGMT is merged with the PD_AON and all reference to the PD_MGMT is replaced by the PD_AON.
SYSSYSCLK	PD_SYS	Gated version of the SYSCLK expected to be used to drive expansion logic that resides in the PD_SYS power domain.
DEBUGDEBUGCLK	PD_DEBUG	Gated version of the DEBUGCLK expected to be used to drive debug expansion logic that resides in the PD_DEBUG power domain. This output clock must exist when HASCSS = 1.
CPUCPU<n>CLK	PD_CPU<n>	Gated version of the CPU<n>CLK expected to be used to drive expansion logic that resides in the PD_CPU<n> powerdomain. when: **PILEVEL = 0** PD_CPU<n> is merged with the PD_SYS and all reference to the PD_CPU<n> is replaced by the PD_SY.
DEBUGCPU<n>CLK	PD_DEBUG	Gated version of the CPU<n>CLK expected to be used to drive debug expansion logic that resides in the the PD_DEBUG power domain. This output clock must exist when HASCSS = 0.

The CRSAS Ma2 provides a Q-Channel interface for each of the output clocks to allow expansion logic to control the availability of each clock output. For more information, see Clock control Q-Channel device interfaces.

6.2 Functional reset inputs and outputs

Reset inputs are used to drive or to request for resets. Reset outputs are used to reset expansion logic that resides in specific power domains.

For more information, see Reset infrastructure.

Unless explicitly stated, it is **IMPLEMENTATION DEFINED** if the following resets are asynchronous or synchronous. We recommend that all resets are asynchronously asserted and synchronously deasserted in relation to the clock of the reset domain, except for reset requests that end with the “REQ” suffix.

The following table lists all the reset interfaces of the subsystem.

Table 6-3: Reset signals

Signal name	Power Domain where it is used when PILEVEL = 2	Direction	Description
nPORESET	PD_AON	Input	Active-LOW Power-On Reset Input.
nSRST	PD_AON	Input	Active-LOW Cold Reset request Input from Debugger.
HOSTRESETREQ	PD_AON	Input	Higher Authority request to perform a Cold reset, for example from a hosting system. This is an asynchronous input. This reset request always results in a system Cold reset regardless of the COLDRESET_MODE configuration. After it is asserted, the signal must be held high until the reset occurs on the nCOLDRESETAON. Holding this input HIGH also results in the system being held in Cold reset.
RESETREQ	PD_AON	Input	Active-HIGH Hardware Request Input to perform a Cold reset. This is an asynchronous input. After it is asserted, the signal must be held HIGH until the reset occurs on the nCOLDRESETAON. The signal must be cleared because of the assertion of the nCOLDRESETAON. When COLDRESET_MODE = 1, this input has no effect on Cold reset directly.
WARMRESETREQ	PD_AON	Input	Active-HIGH Request from expansion logic to perform a Warm reset. This input is associated with the WARMRESETACK and together operate using a four phase handshake. This signal is an asynchronous input. The existence of this signal and the associated acknowledge signal is **IMPLEMENTATION DEFINED**.
WARMRESETACK	PD_AON	Output	Active-HIGH Acknowledge to expansion logic to indicate that a Warm reset has occurred. This input is associated with the WARMRESETREQ. This is an asynchronous output. The existence of this signal and the associated request signal is **IMPLEMENTATION DEFINED**.
nCOLDRESETAON	PD_AON	Output	Active-LOW Cold Reset for the Expansion System. This Cold reset merges other reset sources within the system with the nPORESET to generate this reset in the PD_AON domain. This reset is expected to be used in the PD_AON domain.
nWARMRESETAON	PD_AON	Output	Active-LOW Warm reset for the Expansion System. This Warm reset is generated by combining Warm reset requests from the system along with Cold reset in the PD_AON domain. This reset is expected to be used in the PD_AON domain
nCOLDRESETMGMT	PD_MGMT	Output	Active-LOW Cold reset for the PD_MGMT power domain. This reset is expected to be used in the PD_MGMT domain. When: **PILEVEL = 0** PD_MGMT is merged with the PD_AON and all reference to the PD_MGMT is replaced by the PD_AON. **PILEVEL = 1** PD_MGMT is merged with the PD_AON and all reference to the PD_MGMT is replaced by the PD_AON.

Signal name	Power Domain where it is used when PILEVEL = 2	Direction	Description
nWARMRESETMGMT	PD_MGMT	Output	Active-LOW Warm reset for the PD_MGMT power domain. This reset is expected to be used in the PD_MGMT domain. When: **PILEVEL = 0** PD_MGMT is merged with the PD_AON and all reference to the PD_MGMT is replaced by the PD_AON. **PILEVEL = 1** PD_MGMT is merged with the PD_AON and all reference to the PD_MGMT is replaced by the PD_AON.
nCOLDRESETSYS	PD_SYS	Output	Active-LOW Cold reset for the PD_SYS power domain. This reset is expected to be used in the PD_SYS domain.
nWARMRESETSYS	PD_SYS	Output	Active-LOW Warm reset for the PD_SYS power domain. This reset is expected to be used in the PD_SYS domain.
nCOLDRESETDEBUG	PD_DEBUG	Output	Active-LOW Cold reset for the PD_DEBUG power domain. This reset is expected to be used in the PD_DEBUG domain. This output must exist when HASCSS = 1.
nWARMRESETCPU<n>	PD_CPU<n>	Output	Active-LOW Warm reset for each of the PD_CPU<n> power domain. These resets are expected to be used in the PD_CPU<n> domains. When: **PILEVEL = 0** PD_CPU<n> is merged with the PD_SYS and all reference to the PD_CPU<n> is replaced by the PD_SYS.
nCOLDRESETCPU<n>	PD_CPU<n>	Output	Active-LOW Cold reset for each of the PD_CPU<n> power domain. These resets are expected to be used in the PD_CPU<n> domains. When: **PILEVEL = 0** PD_CPU<n> is merged with the PD_SYS and all reference to the PD_CPU<n> is replaced by the PD_SYS.
nCOLDRESETDEBUGCPU<n>	PD_DEBUG	Output	Active-LOW Cold reset for PD_DEBUG associated with the debug logic of each processor. These are expected to be used in the PD_DEBUG domain.

6.3 P-Channel and Q-Channel Device interfaces

Each P-Channel or Q-Channel Device interface independently allows external expansion logic to handshake with the system to ensure that:

For warm reset control

All external managers and subordinate interfaces in a Warm reset domain are in safe state and isolated from the Cold reset domain before asserting Warm reset. It also allows the external manager to issue a wake request to a power domain.

For clock control

All external dependent logic can prepare itself before the hierarchical gating of clocks.

During system integration, expansion logic that resides within a power or clock domain associated with each P-Channel or Q-Channel normally merges all P-Channel or Q-Channel interfaces within the expansion domain to drive each interface.

When merging, the following rules must be obeyed to prevent system deadlocks:

- All bus managers in the expansion system must, either individually or collectively have a full Q-Channel interface, or a P-Channel interface.

  Each Q-Channel interface must be able to deny a quiescence request if the logic that it controls has outstanding operations on the bus or is unable to enter quiescent state for any other reason. Similarly, each P-Channel interface must be able to deny a request to enter a different PSTATE if the logic is unable to enter the requested state. The exception is the WARM_RST power state where managers are expected to suspend their activity cleanly, activate internal reset domain crossing protections and always accept the request.

  Each P-Channel interface for power control must be able to support all power states that the domain implements.

- All bus subordinates in the expansion system must, either individually or collectively, have a Q-Channel or P-Channel interface that must be able to delay the acceptance of a quiescence request or a Power state request if the current bus operation is about to complete.

  The LPI interface must also be able to deny a quiescence request or a power state request if the logic that it controls has other outstanding operations that prevent it from entering quiescent or the requested state. The exception is the WARM_RST power state where subordinates are expected to suspend their activity cleanly, activate internal reset domain crossing protections and always accept the request.

- You must sequence the Q-Channels associated with these external bus interfaces to ensure that all bus managers are in quiescent state before any bus subordinates are requested to enter quiescent state.

  Similarly, with P-Channels, you must ensure that all bus managers and subordinate of these interfaces can enter the new requested state in the right order according to their dependencies before the P-Channel accepts entering that state.

  An implementation of the system can depend on external sequencing to be added by a system integrator to do this for their expansion logic, or alternatively, implement multiple Q-Channels or P-Channel interfaces for each domain to separately handshake managers, subordinates, and even intermediate components in the domain in sequence.

If a Q-Channel Device interface is not used, its associated QACTIVE and QDENY signals must be tied LOW and the QREQn output looped back into its QACCEPTn input. Similarly, if a P-Channel Device interface is not used, then its associated PACTIVE and PDENY must be tied LOW, and the PREQn output looped back into its PACCEPTn input.

Note

Not used means there is no consumer of the corresponding clock or power, this not to be confused with the case where the consumer of the corresponding clock or power has no LPI interface.

Individual signals of the Q-Channel Device interfaces are sometimes described as a vector. For example, a Q-Channel Device interface could have *x* bits for each signal. In this case, bit *i* of all signals within *0* to *x-1* forms a single bit Q-Channel interface. For example, QACTIVE[1], QREQn[1], QACCEPTn[1] and QDENY[1] form a Q-Channel interface and there are x Q-Channel interfaces. When a Q-Channel device interface is defined as a vector in this document, unless otherwise stated, it means the following:

- QACTIVE[x-1:0] are considered as OR-ed together at the controller,
- Collectively all of the *i* Q-Channel interfaces have to enter Q_STOPPED state before the entire interface is considered to be in the Q_STOPPED state. And similarly, all Q-Channel *i* have to enter Q_RUN state before the entire interface is considered to be in the Q_RUN state.
- When transitioning the entire interface between Q_RUN state to Q_STOPPED state, if any of the individual bit *i* Q-Channel interfaces denies the transition, all other Q-Channel must also return to the original Q-Channel state.

When P-Channel Device interface is used, the P-Channel encoding replicates the PCK-600 PPU's Device P-Channel bit assignment, with each DEVPACTIVE bit used to request entry to a Power mode and Operating mode, and the DEVPSTATE vector representing the power mode being requested.

For more information, see *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual* and *Arm® Power Policy Unit Architecture Specification*.

For more information on the Q-Channel and P-Channel protocol, see *AMBA® Low Power Interface Specification - Arm® Q-Channel and P-Channel Interfaces*.

6.3.1 Clock control Q-Channel device interfaces

The CRSAS Ma2 defines a Q-Channel Device interface for each of the output clocks to allow expansion logic to control the availability of each clock output. These are used to support high-level clock gating. Each interface can either be single bit or a vector, and is **IMPLEMENTATION DEFINED**.

The following Q-Channels are provided:

- MGMTSYSCLK Q-Channel Device interface for MGMTSYSCLK. This interface resides in the PD_MGMT power domain when PILEVEL = 2, or in the PD_AON domain when PILEVEL < 2.
- SYSSYSCLK Q-Channel Device interface for SYSSYSCLK. This interface resides in the PD_SYS power domain.
- DEBUGDEBUGCLK Q-Channel Device interface for DEBUGDEBUGCLK. This interface must exist when HASCSS = 1. This interface resides in the PD_DEBUG power domain.

- CPUCPU<n>CLK Q-Channel Device interface for CPUCPU<n>CLK. This interface resides in the PD_CPU<n> power domain when PILEVEL > 0, or in the PD_SYS power domain when PILEVEL = 0.
- DEBUGCPU<n>CLK Q-Channel Device interface for DEBUGCPU<n>CLK. This interface must exist when HASCSS = 0. This interface resides in the PD_DEBUG power domain when PILEVEL > 0, or in the PD_SYS power domain when PILEVEL = 0.

All clock Q-Channel Device interfaces are for clock control only and do not support waking the system from hibernation.

Each of the Clock Control Device Q-Channel interfaces can be multi-bit and are either an asynchronous interface or synchronous to the clock that each of the Q-Channel interfaces control. The choice is **IMPLEMENTATION DEFINED**.

6.3.2 Power Control Q-Channel and P-Channel Expansion Device interfaces

The CRSAS Ma2 provides power control Q-Channel or P-Channel Device interfaces to allow expansion logic to handshake and coordinate the expansion logic power state.

Each power domain that supports expansion is provided with either Q-Channel or P-Channel interfaces as follows:

- AONPWR Q-Channel or P-Channel Device interface for PD_AON. This interface resides in the PD_AON power domain and can be tied or used to handshake Warm reset entry for expansion logic residing in the PD_AON domain. When MGMT_PPU or PD_MGMT exist, AONPWR Q-Channel or P-Channel Device interface is optional and MGMTPWR Q-Channel can be used instead for handshaking Warm reset entry for expansion logic residing in the PD_AON domain.
- MGMTPWR Q-Channel or P-Channel Device interface for PD_MGMT. This interface resides in the PD_MGMT power domain and must exist when PILEVEL = 2. When PILEVEL < 2, this interface is not required, but if it exists, it resides in PD_AON domain and can be tied or used to handshake Warm reset entry for expansion logic residing in the PD_MGMT domain.
- SYSPWR Q-Channel or P-Channel Device interface for PD_SYS. This interface resides in the PD_SYS power domain
- DEBUGPWR Q-Channel or P-Channel Device interface for PD_DEBUG. This interface resides in the PD_DEBUG power domain and must exist when PILEVEL > 0. When PILEVEL = 0, this interface is not required, but if it exists, it resides in PD_SYS domain and can be used for debug logic that now resides in PD_SYS, or must be loop backed and tied.
- CPU<n>PWR Q-Channel or P-Channel Device interface for PD_CPU<n>. This interface must exist when PILEVEL > 0. When PILEVEL = 0, this interface is not required, but if it exists, it resides in PD_SYS domain and must be loop backed and tied.

The PD_NPU is not provided with Q-Channel or P-Channel interfaces as it does not support expansion.

For more information, especially on the encoding of the PSTATE and PACTIVE signals of the above P-Channel interfaces, see *Arm® Power Policy Unit Architecture Specification* and *Arm® CoreLink™ PCK-600 Power Control Kit Technical Reference Manual*.

These Q-Channel or P-Channel Device interfaces are driven by expansion logic that resides within their respective power domain that they control. These Q-Channel and P-Channel Device interfaces are used by a variety of entities to perform their respective operations:

- Used by the expansion logic to indicate through the QACTIVE or PACTIVE signal that the expansion logic is IDLE or indicate that it wants to enter a different power state.
- For a power controller to request the expansion logic to enter a different power state.
- Allow the expansion logic to accept or deny the request to enter a different power state.

The choice between Q-Channel or P-Channel interface is **IMPLEMENTATION DEFINED**. The Q-Channel or P-Channel interfaces do not support the waking of the power domain, since they reside within the power domain that is being controlled. Instead, associated Power Control Wakeup Q-Channel Device interfaces described in Power Control Wakeup Q-Channel Device interfaces should be used to request for specific power domains to power up.

Currently, each domain is described as having at least one Q-Channel or P-Channel interface. However, an implementation can provide multiple Q-Channel, multi-bit Q-Channel or multiple P-Channel interfaces per domain in order, for example, to sequence expansion logic when entering lower power states. The number of Power Control Q-Channels or P-Channels per domain is therefore **IMPLEMENTATION DEFINED**.

These Power Control Device Q-Channel or P-Channel interfaces are either asynchronous interfaces or are synchronous to the clock used in each domain that each of the Q-Channel or P-Channel interfaces control. The choice is **IMPLEMENTATION DEFINED**.

6.3.3 Power Control P-Channel Expansion Device interfaces

The CRSAS Ma2 provides power control P-Channel Device interfaces to allow expansion of the existing power domain hierarchy with new power domains by coordinating allowed power states of external power domain in the power domain hierarchy.

Each power domain that supports a power domain hierarchy extension is provided with a P-Channel interface as follows:

MGMTPDHCPWR P-Channel Device interface for PD_MGMT

This interface resides in the PD_MGMT power domain when PILEVEL = 2. When PILEVEL < 2, this interface resides in PD_AON power domain. The interface must only be used for handshaking and coordinating allowed power states of external power domains which are meant to be at the same level as PD_SYS and PD_DEBUG in the power domain hierarchy. The interface is running on MGMTSYSCLK and reset on nCOLDRESETMGMT.

SYSPDHCPWR P-Channel Device interface for PD_SYS

This interface resides in the PD_MGMT power domain when PILEVEL = 2. When PILEVEL < 2, this interface resides in PD_AON power domain. The interface must only be used for handshaking and coordinating allowed power states of external power domains which are meant to be at the same level as PD_NPU and PD_CPU in the power domain hierarchy. The interface is running on MGMTSYSCLK and reset on nCOLDRESETMGMT.

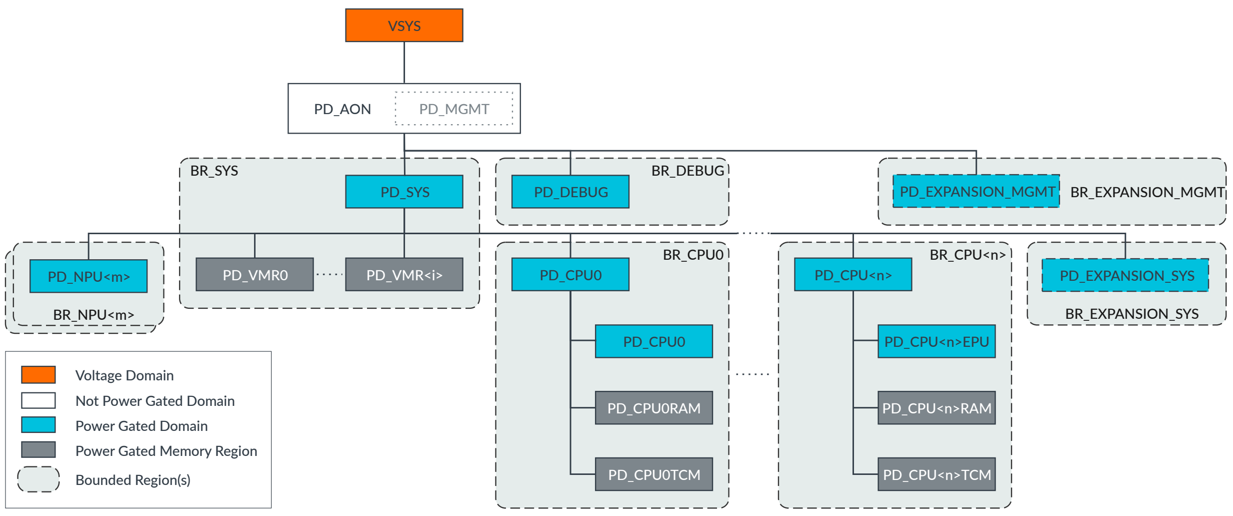
The following figure shows an example power domain heirarchy where two expansion power domains are added. The MGMTPDHCPWR P-Channel Device interface is be used for power coordination with BR_EXPANSION_MGMT which controls PD_EXPANSION_MGMT power domain while SYSPDHCPWR P-Channel Device interface is used for power coordination with BR_EXPANSION_SYS that controls PD_EXPANSION_SYS.

Currently, each domain is described as having a minimum of a single P-Channel interface. However, an implementation can provide multiple P-Channel interfaces per domain. For example, to enable multiple power domains to be added and controlled. The number of Power Control P-Channel interfaces per domain is therefore **IMPLEMENTATION DEFINED**.

These Device P-Channel interfaces are either an asynchronous interface or are synchronous to the clock used in each domain that each of the P-Channel interface controls. The choice is **IMPLEMENTATION DEFINED**.

For more information on power and voltage domains, please see the following figure and Power Control Infrastructure.

Figure 6-1: Power control P-Channel expansion device interfaces with PILEVEL = 1



6.3.4 Power Control Wakeup Q-Channel Device interfaces

The CRSAS Ma2 provides Power Control Wakeup Q-Channel Device interfaces to allow expansion logic to request power up of specific domains. The Q-Channel interfaces and the domains they control are:

- PWRMGMTWAKE Q-Channel Device interface for PD_MGMT. This interface must exist when PILEVEL = 2.
- PWRSYSWAKE Q-Channel Device interface for PD_SYS.
- PWRDEBUGWAKE Q-Channel Device interface for PD_DEBUG.

- PWRCPU<n>WAKE Q-Channel interface for PD_CPU<n>. This interface must exist when PILEVEL > 0.

These interfaces all reside in the PD_AON domain. Because they are power up requests, also referred to as wake up request, we recommend that at minimum, only the QACTIVE signal of each interface is implemented. Therefore, when using these to wake a domain, you must drive and hold the appropriate QACTIVE signal to request to turn on the domain until the domain is ON. For example, to wake PD_SYS, you must set PWRSYSWAKEQACTIVE Q-Channel signal to request to turn ON until the SYSPWR Q-Channel or P-Channel Device Interface for PD_SYS indicates that the power domain is ON.

These Wakeup Device Q-Channel interfaces are either asynchronous or are synchronous to the clock used in each domain that each of the Q-Channel interfaces control. The choice is **IMPLEMENTATION DEFINED**.

6.4 Clock Control Q-Channel Control interfaces

Each Q-Channel Control interface in this section is single bit per signal on the Q-Channel interface and independently allows the system to request for the availability of a clock source. Each Q-Channel Control interface allows an external clock controller to handshake with the system to safely turn the clock source OFF.

Each of the clock control Q-Channel interfaces can be either asynchronous or synchronous to the clock that each of the Q-Channel interfaces control. The choice is **IMPLEMENTATION DEFINED**.

The CRSAS Ma2 has the following clock control Q-Channel Control interfaces:

- AONCLK Q-Channel Control interface for AONCLK.
- SYSCLK Q-Channel Control interface for SYSCLK.
- CPU<n>CLK Q-Channel Control interface for CPU<n>CLK.
- NPU<m>CLK Q-Channel Control interface for NPU<m>CLK.
- DEBUGCLK Q-Channel Control interface for DEBUGCLK. This interface must exist if HASCSS = 1.

If an input clock source is always running, and there is no clock controller associated with this clock input, then you can tie its associated clock control Q-Channel Control interface by tying QREQn input to HIGH. These interfaces are in the PD_AON domain.

6.5 Expansion Power Control Dependency interface

The CRSAS Ma2 provides an optional set of 2, up to 4-bit width Q-Channel interfaces that allow external power domains to use the Power Dependency Control Matrix to keep power domains within the subsystem from entering a lower power state.

These are asynchronous signals that either reside in the PD_AON power domain or in the PD_MGMT domain, and the choice is **IMPLEMENTATION DEFINED**. For more information on registers that use these signals and description of related functionality, see System Control Register block and Power Control Infrastructure.

The signals are as follows:

PDCMONQREQn[<PDCMQCHWIDTH-1>:0]

Power Dependency Control Matrix QREQn inputs.

When each bit is set to '1', it indicates that the external domain that drives it is in functional power mode.

PDCMONQACCEPTn[<PDCMQCHWIDTH-1>:0]

Power Dependency Control Matrix QACCEPTn outputs.

Each bit acknowledges an associated bit on PDCMONQREQn by returning the request input value. This acts as a four-phase handshake so that the driver of each request bit can determine that the request has been seen by receiving unit.

PDCMRETQREQn[<PDCMQCHWIDTH-1>:0]

Power Dependency Control Matrix QREQn inputs.

When each bit is set to '1', it indicates that the external domain that drives it is in functional power mode.

PDCMRETQACCEPTn[<PDCMQCHWIDTH-1>:0]

Power Dependency Control Matrix QACCEPTn outputs.

Each bit acknowledges an associated bit on PDCMRETQREQn by returning the request input value. This acts as a four-phase handshake so that the driver of each request bit can determine that the request has been seen by receiving unit.

These signals provides a way for an external power domain to request for a domain within the system to remain powered so that the external domain can access the internal power domain or at least request that the internal domain retain its register context. For example, the external domain may want to access the Main interconnect in the PD_SYS domain.

1. External domain raises an interrupt with the host processor, which, if not already ON, wakes up PD_SYS.
2. Host processor sets the relevant bit in PDCM_PD_SYS_SENSE.S_PDCMONQREQ{0-<PDCMQCHWIDTH-1>}.

The external system can now keep PD_SYS powered using one of the PDCMONQREQn signals.

Similarly, if the external domain required PD_SYS to maintain state, but did not currently require access to it:

1. External domain raises an interrupt with the host processor, which if not already ON, wakes up PD_SYS.
2. Host processor sets the relevant bit in PDCM_PD_SYS_SENSE.S_PDCMRETQREQ{0-<PDCMQCHWIDTH-1>}.

The external system can now ensure PD_SYS retains state using one of the PDCMRETQREQn signals.

The four-phase handshake must be used to ensure that there is no race condition between ending a bus access that wakes-up the domain and activating the keep-up of the domain.

6.6 Power Domain ON Status signals

The CRSAS Ma2 provides a set of output signals that indicates if the power domain each is associated with is in the ON Power Mode.

These signals are:

PDMGMTON

- HIGH indicates that the PPU of the PD_MGMT power domain presumes that the domain is ON
- LOW indicates that the PPU presumes that the power domain is in a lower power state
- This signal must exist when PILEVEL = 2.

PDSYSON

- HIGH indicates that the PPU of the PD_SYS power domain presumes that the domain is ON
- LOW indicates that the PPU of the power domain presumes that the domain is in a lower power state.

PDCPU<n>ON

- HIGH indicates that the PPU of the PD_CPU<n> power domain presumes that the domain is ON
- LOW indicates that the PPU of the power domain presumes that the domain is in a lower power state
- This signal must exist when PILEVEL > 0.

PDNPU<m>ON

- HIGH indicates that the PPU of the PD_NPU<m> power domain presumes that the domain is ON
- LOW indicates that the PPU of the power domain presumes that the domain is in a lower power state
- This signal only exists if NUMNPU > 0.

PDDEBUGON

- HIGH indicates that the PPU of the PD_DEBUG power domain presumes that the domain is ON
- LOW indicates that the PPU of the power domain presumes that the domain is in a lower power state

Note

These are primarily status signals and are typically driven using PPUHWSTAT signals. We recommend that these are not used as power control signals directly.

6.7 System Timestamp interface

The CRSAS Ma2 provides a System Timestamp input from an expansion timestamp counter. This timestamp is expected to be driven by a timestamp generator in the subsystem expansion. This resides in the PD_AON power domain and nWARMRESETAON reset domain.

The System Timestamp interface has the following properties:

- Name: CNTVALUE<G/B>[{23-63}:0]
- Description: Timestamp input value. This value can either be:
    - Gray coded. In this case, the signal name ends with 'G' and is asynchronous.
    - Binary coded. In this case, the signal name ends with 'B' and is synchronous to CNTCLK.
- Width: 24-64 bit, IMPL_DEF
- Direction: Input
- Clock domain: CNTCLK or is asynchronous

When HASCSS = 1, we recommend that the expansion system uses the CTI triggers to implement timestamp halting. See sections Cross Trigger Interface and Cross Trigger for more information on the CTI interface.

When HASCSS = 0, a Debug System does not exist to provide these control signals. Therefore, the system integrator has to depend on the CPU<n>HALTED signals to halt the System Timestamp generator.

6.8 Main Interconnect Expansion interfaces

The CRSAS Ma2 provides a configurable number of Manager and Subordinate Expansion interfaces of the Main Interconnect. These interfaces allow the system integrator to add additional bus managers and bus subordinates to the system.

For more information, see System interconnect infrastructure.

The AMBA protocol used for this interface can either be AHB5 or AXI5 and must at least support the following properties:

- 32-bit address
- 32-bit, 64-bit, or 128-bit data
- Synchronous to SYSSYSCLK

- On the nWARMRESETSYS reset
- TrustZone Support enabled

We recommend that these interfaces are 64-bit or 128-bit wide and use the AXI5 protocol.

The types of Manager and Subordinate Expansion interfaces on the Main Interconnect are as follows:

Manager Code Main Expansion interface

This interface provides access to Code Memory and is mapped to the following address range:

- `0x0100_0000` to `0x09FF_FFFF`
- `0x1100_0000` to `0x19FF_FFFF`, with a 16MB region from `0x1100_0000` to `0x11FF_FFFF` mapped alternatively to the Boot Rom if and only if the Boot Rom exist.

There must be at least one such interface in the subsystem. If there are more than one such interfaces, it is **IMPLEMENTATION DEFINED** how the preceding address range is divided across the interfaces. If any implemented interface is not used or any of the preceding region is not implemented, a default subordinate must be used to respond with a bus error. This interface must export information to allow a debug access to be distinguished from an access by another manager in the system. We recommend that one such interface is provided in an implementation of the subsystem.

Manager Main Expansion interface

This interface provides access to other subordinates in the system and is mapped to the following address range:

- `0x2800_0000` to `0x2FFF_FFFF`
- `0x3800_0000` to `0x3FFF_FFFF`
- `0x6000_0000` to `0xDFFF_FFFF`

There can be zero or more such interfaces in the subsystem. If there are no such interfaces, all access to the above address range results in decode error. Additionally, if any implemented interface is not used or any of the preceding regions is not implemented, a default subordinate must be used to respond to the access targeting the interface and any address region not implemented with decode error. If there are more than one such interfaces, it is **IMPLEMENTATION DEFINED** how the preceding address range is divided across the interfaces. If implemented, all these interfaces must export information to allow a debug access to be distinguished from an access by another manager in the system. We recommend that one such interface is provided in an implementation of the subsystem.

Subordinate Main Expansion interface

This interface provides access to system from expansion managers. This interface can be used to access all memory mapped regions in the system except for regions private to the processors. We recommend that one such interface is provided in an implementation of the subsystem. If any implemented interface is not used, the interface must be tied to idle.

6.9 Peripheral Interconnect Expansion interfaces

The CRSAS Ma2 provides a configurable number of Manager and Subordinate Expansion interfaces from the Peripheral Interconnect. These interfaces allow the system integrator to add additional bus managers and bus subordinates to the system that is expected to require lower latency access to peripherals.

For more information, see System interconnect infrastructure.

The AMBA protocol used for this interface can either be AHB5 or AXI5 and must at least support the following properties:

- 32-bit address
- 32-bit or 64-bit data
- Synchronous to SYSSYSCLK
- On the nWARMRESETSYS reset
- TrustZone Support enabled

We recommend that these interfaces are 32-bit wide and use the AHB5 protocol.

The types of Manager and Subordinate Expansion interfaces on the Peripheral Interconnect are as follows:

Manager Peripheral Expansion interface

This interface provides access to other subordinates in the system and is mapped to the following address range:

- `0x4010_0000` to `0x47FF_FFFF`
- `0x4810_0000` to `0x4FFF_FFFF`
- `0x5010_0000` to `0x57FF_FFFF`
- `0x5810_0000` to `0x5FFF_FFFF`
- `0xE020_0000` to `0xEFFF_FFFF`
- `0xF020_0000` to `0xFFFF_FFFF`

There can be zero or more such interfaces in the subsystem. If there are no such interfaces, all access to the preceding address range it targets responds with bus error. Additionally, if any implemented interface is not used or any of the preceding region is not implemented, a default subordinate must be used to respond with decode error. If there are more than one such interfaces, it is **IMPLEMENTATION DEFINED** how the preceding address range is divided across the interfaces. If implemented all these interfaces must export information to allow a debug access to be distinguished from an access by another manager in the system. We recommend that one such interface is provided in an implementation of the subsystem.

Subordinate Peripheral Expansion interface

This interface provides access to the Peripheral bus from expansion managers. This interface can be used to access all memory mapped regions in the system except to regions private to the processors. However, we recommend that this interface is not used to access areas outside the following memory mapped regions because of potential memory throughput limitations because of bus protocol and bus width conversion incurred at implementation:

- `0x4000_0000` to `0x4FFF_FFFF`
- `0x5000_0000` to `0x5FFF_FFFF`
- `0xE010_0000` to `0xEFFF_FFFF`
- `0xF010_0000` to `0xFFFF_FFFF`

We recommend that one such interface is provided in an implementation of the subsystem. If any implemented interface is not used, the interface must be tied to idle.

6.10 Interrupt interfaces

The CRSAS Ma2 includes interrupt signals for use by the subsystem expansion. These connect to the interrupt controller of each processor within the system and optionally to an External Wakeup Controller (EWIC) associated with the processor or the Internal Wakeup Interrupt Controller (IWIC) of the processor.

Signal name	Width	Direction	Description
CPU<n>EXPIRQ[CPU<n>EXPNUMIRQ-1:0]	CPU<n>EXPNUMIRQ	Input	These are interrupt inputs from the subsystem expansion to the CPU<n> interrupt controller within the subsystem. Each processor in the subsystem implements a configurable number of external interrupt lines. 32 of the external interrupt lines are reserved for internal use and the rest are made available here. CPU<n>EXPNUMIRQ defines the number of interrupts made available as expansion interrupts for CPU<n>.
CPU<n>EXPNMI	1	Input	This provides a non-maskable interrupt input from the subsystem expansion to the interrupt controller of CPU<n> within the subsystem. This input is merged with other non-maskable interrupt sources within the subsystem before it is seen by the NVIC of the processor core.

Note

Each bit CPU<n>EXPIRQ[i] is ultimately connected to IRQ[32+i] of the CPU<n>'s NVIC.

6.11 DMA interfaces

The CRSAS Ma2 supports a DMA in the system. When DMA exists, namely NUMDMA > 1, the following interfaces can exist fully or partially, depending on the configuration of the DMA:

- DMA Trigger interfaces
- DMA General Purpose Output (GPO) interfaces
- DMA Status and Control Interfaces (except halt and restart CTI interface)
- DMA Stream interfaces
- DMA Individual Channel Interrupts
- DMA Boot Configuration signals

For more information on DMA-350, see *Arm® CoreLink™ DMA-350 Controller Technical Reference Manual*.

If any DMA interfaces exist, they all reside in the PD_SYS power domain, run on SYSSYSCLK, and they are on the nWARMRESETSYS reset domain.

6.12 CPU Coprocessor Interface

Each processor core of the subsystem can be configured to have a coprocessor interface. If a CPU<*n*> coprocessor interface exists, then HASCPU<*n*>CPIF = 1.

These interfaces reside in their respective processor's PD_CPU<*n*> power domain, CPUCPU<n>CLK clock domain and nWARMRESETCPU<n> reset domain. Note that because of the actual processor implementation, the reset can be a special output that is dependent at least on nWARMRESETCPU<n>.

For more information on the coprocessor and related interfaces, see *Arm® Cortex®-M55 Processor Integration and Implementation Manual* or *Arm® Cortex®-M85 Processor Integration and Implementation Manual*.

6.13 TCM subordinate interface

A 64-bit subordinate TCM interface provides system access only to Tightly Coupled Memories (TCM) internal to each CPU. The protocol of this interface is **IMPLEMENTATION DEFINED**.

This expansion interface is typically used together with DMA controllers to transfer data to and from the processors for compute applications and each provides access to the TCM DMA interface. For example, for a Cortex M55 or Cortex-M85 core this TCM DMA interface is called the S-AHB DMA interface.

For more information on S-AHB or TCM-AHB DMA interface, see *Arm® Cortex®-M55 Processor Integration and Implementation Manual* or *Arm® Cortex®-M85 Processor Integration and Implementation Manual*.

This interface supports the following properties:

- 32-bit address
- 64-bit data for Cortex-M55 and Cortex-M85
- Normal Memory access only
- Additional byte lane strobe signals to support non-contiguous write-data in a beat
- TrustZone Support enabled

The memory map presented on each of these interfaces and the underlying functionality for TCM access is defined by the *Arm® Cortex®-M55 Processor Technical Reference Manual* or *Arm® Cortex®-M85 Processor Technical Reference Manual*.

The memory map is shown in the following table. Instruction TCM (ITCM) accesses are mapped into a single address space, while each 64-bit Data TCM (DTCM) access is mapped to two 32-bit wide DTCMs, with Cortex-M55 and Cortex-M85 supporting four 32-bit wide DTCMs.

Each of these interfaces resides either in the PD_SYS power domain or in their respective processor's PD_CPU<*n*> power domain. Depending on where they reside, the following conditions are true:

- If in the PD_CPU<*n*> power domain, the interface is on CPUCPU<n>CLK clock domain and nWARMRESETCPU<n> reset domain.
- If in the PD_SYS power domain, the interface is on SYSSYSCLK clock domain and nWARMRESETSYS reset domain.

Note

These addresses in the previous table are specific only for the TCM DMA subordinate interface. These TCMs reside at address offsets in the main memory map and at private address offsets to each processor's local TCM. To make these TCMs visible to other managers outside of the subsystem, and also to other CPUs, each CPU TCMs are aliased to a different location on the memory map. For more information, see CPU TCM memories.

Global exclusive access monitors are not supported on TCM Subordinate interface. Ensure that no coherency issues arise when you are using this interface to access the TCMs. It is valid for processors in the system to cache TCM instruction and data values. Therefore, care must be taken to ensure consistency of the TCM and Cache data.

Table 6-5: TCM DMA interface memory map

Start address	End address	xADDRS[3:2]	TCM accessed for Cortex-M55 and Cortex-M85
`0x0A00_0000`	`0x0A00_0000` + {ITCM size}	-	Non-secure CPU0 ITCM
`0x0B00_0000`	`0x0B00_0000` + {ITCM size}	-	Non-secure CPU1 ITCM
`0x0C00_0000`	`0x0C00_0000` + {ITCM size}	-	Non-secure CPU2 ITCM
`0x0D00_0000`	`0x0D00_0000` + {ITCM size}	-	Non-secure CPU3 ITCM
`0x1A00_0000`	`0x1A00_0000` + {ITCM size}	-	Secure CPU0 ITCM

Start address	End address	xADDRS[3:2]	TCM accessed for Cortex-M55 and Cortex-M85
0x1B00_0000	0x1B00_0000 + {ITCM size}	-	Secure CPU1 ITCM
0x1C00_0000	0x1C00_0000 + {ITCM size}	-	Secure CPU2 ITCM
0x1D00_0000	0x1D00_0000 + {ITCM size}	-	Secure CPU3 ITCM
0x2400_0000	0x2400_0000 + {DTCM size}	2b00	Non-secure CPU0 D0TCM
0x2400_0000	0x2400_0000 + {DTCM size}	2b01	Non-secure CPU0 D1TCM
0x2400_0000	0x2400_0000 + {DTCM size}	2b10	Non-secure CPU0 D2TCM
0x2400_0000	0x2400_0000 + {DTCM size}	2b11	Non-secure CPU0 D3TCM
0x2500_0000	0x2500_0000 + {DTCM size}	2b00	Non-secure CPU1 D0TCM
0x2500_0000	0x2500_0000 + {DTCM size}	2b01	Non-secure CPU1 D1TCM
0x2500_0000	0x2500_0000 + {DTCM size}	2b10	Non-secure CPU1 D2TCM
0x2500_0000	0x2500_0000 + {DTCM size}	2b11	Non-secure CPU1 D3TCM
0x2600_0000	0x2600_0000 + {DTCM size}	2b00	Non-secure CPU2 D0TCM
0x2600_0000	0x2600_0000 + {DTCM size}	2b01	Non-secure CPU2 D1TCM
0x2600_0000	0x2600_0000 + {DTCM size}	2b10	Non-secure CPU2 D2TCM
0x2600_0000	0x2600_0000 + {DTCM size}	2b11	Non-secure CPU2 D3TCM
0x2700_0000	0x2700_0000 + {DTCM size}	2b00	Non-secure CPU3 D0TCM
0x2700_0000	0x2700_0000 + {DTCM size}	2b01	Non-secure CPU3 D1TCM
0x2700_0000	0x2700_0000 + {DTCM size}	2b10	Non-secure CPU3 D2TCM
0x2700_0000	0x2700_0000 + {DTCM size}	2b11	Non-secure CPU3 D3TCM
0x3400_0000	0x2400_0000 + {DTCM size}	2b00	Secure CPU0 D0TCM
0x3400_0000	0x3400_0000 + {DTCM size}	2b01	Secure CPU0 D1TCM
0x3400_0000	0x3400_0000 + {DTCM size}	2b10	Secure CPU0 D2TCM
0x3400_0000	0x3400_0000 + {DTCM size}	2b11	Secure CPU0 D3TCM
0x3500_0000	0x3500_0000 + {DTCM size}	2b00	Secure CPU1 D0TCM
0x3500_0000	0x3500_0000 + {DTCM size}	2b01	Secure CPU1 D1TCM
0x3500_0000	0x3500_0000 + {DTCM size}	2b10	Secure CPU1 D2TCM
0x3500_0000	0x3500_0000 + {DTCM size}	2b11	Secure CPU1 D3TCM
0x3600_0000	0x3600_0000 + {DTCM size}	2b00	Secure CPU2 D0TCM
0x3600_0000	0x3600_0000 + {DTCM size}	2b01	Secure CPU2 D1TCM
0x3600_0000	0x3600_0000 + {DTCM size}	2b10	Secure CPU2 D2TCM
0x3600_0000	0x3600_0000 + {DTCM size}	2b11	Secure CPU2 D3TCM
0x3700_0000	0x3700_0000 + {DTCM size}	2b00	Secure CPU3 D0TCM
0x3700_0000	0x3700_0000 + {DTCM size}	2b01	Secure CPU3 D1TCM
0x3700_0000	0x2700_0000 + {DTCM size}	2b10	Secure CPU3 D2TCM
0x3700_0000	0x2700_0000 + {DTCM size}	2b11	Secure CPU3 D3TCM

6.14 Debug and Trace related interfaces

This section describes the debug and trace related interfaces of the CRSAS Ma2.

6.14.1 Debug Access interface

When DEBUGLEVEL > 0, the CRSAS Ma2 provides interfaces for debug access from an external DAP or an external debug infrastructure. Depending on the HASCSS configuration, the interface or interfaces provide the following:

- When HASCSS = 0, where there can only be one processor core, the CPU Debug D-AHB access is provided as an expansion interface. This allows the SoC integrator to drive the interface using a suitable CoreSight MEM-AP and provides debug access to the processor. For more information on the processor's D-AHB interface, see *Arm® Cortex®-M55 Processor Technical Reference Manual* or *Arm® Cortex®-M85 Processor Technical Reference Manual*.

  The interface is in PD_CPU0 power domain and resides in the CPUCPU0CLK clock domain and nCOLDRESETCPU0 reset domain for Cortex-M55, or nCOLDRESETDEBUGCPU0 and DEBUGCPU0CLK for Cortex-M85.
- When HASCSS = 1, a single Debug Access Interface is provided as an expansion subordinate interface. This interface is an APB4 subordinate interface and provides an additional DPABORT input signal to allow an external DAP to signal a transaction abort. This interface resides in the DEBUGDEBUGCLK clock domain and nCOLDRESETDEBUG reset domain.

When DEBUGLEVEL = 0, these interfaces might not exist. If they do exist, they are tied or unused.

6.14.2 Debug Timestamp interface

When DEBUGLEVEL = 2, the CRSAS Ma2 provides one or more 64-bit timestamp input or inputs. This timestamp is expected to be driven by a timestamp generator in the subsystem expansion. Depending on HASCSS configuration, the interface or interfaces are as follows:

- When HASCSS = 0 where there can only be one processor core, the processor's debug global timestamp input, TSVALUEB[63:0], is provided as an expansion interface, CPU0TSVALUEB[63:0]. This is expected to be driven by a global timestamp generator. For more information on these interfaces, see *Arm® Cortex®-M55 Processor Technical Reference Manual* or *Arm® Cortex®-M85 Processor Technical Reference Manual*.

  This interface resides in the PD_DEBUG power domain and resides in the DEBUGCPU0CLK clock domain and nCOLDRESETDEBUGCPU0 reset domain.
- When HASCSS = 1, a single TSVALUE<B/G> is provided and is used to generate all debug timestamps input to all processors within the system. It resides in the PD_DEBUG power domain. The Timestamp input value TSVALUE<B/G> width is **IMPLEMENTATION DEFINED**, and it can either be:
  - Gray coded. In this case, the signal name ends with 'G' and is asynchronous.
  - Binary coded. In this case, the signal name ends with 'B' and is synchronous to DEBUGDEBUGCLK and is on the nCOLDRESETDEBUG reset domain.

For both configurations of HASCSS, each processor's TSCLKCHANGE input is provided as an expansion interface as CPU<n>TSCLKCHANGE to allow the CPUs to be notified of a change in timestamp clock ratio. Each interface that is associated with CPU<n> resides in their respective DEBUGCPU<n>CLK clock domain and nCOLDRESETDEBUGCPU<n> reset domain.

When DEBUGLEVEL < 2, this timestamp interface might not exist. If they do exist, they are tied or unused.

6.14.3 Cross Trigger Channel interface

When DEBUGLEVEL > 0, the CRSAS Ma2 includes one or more sets of Cross Trigger Channel inputs and Cross Trigger Channel outputs to allow partners to expand the cross trigger infrastructure as follows.

When HASCSS = 1, the CRSAS Ma2 includes a shared Cross Trigger Matrix (CTM) and provides a single Cross Trigger Channel input and a single Cross Trigger Channel output to allow partners to expand the cross trigger infrastructure. See the table below.

For more information on the CTM, see *Arm® CoreSight™ System-on-Chip SoC-600M Technical Reference Manual*. This interface is synchronous to DEBUGDEBUGCLK, resides in the PD_DEBUG power domain and nCOLDRESETDEBUG reset domain.

Table 6-6: Cross Trigger Channel interface

Signal name	Width	Direction	Description
CTMCHANNELIN[3:0]	4	Input	Channel in port
CTMCHANNELOUT[3:0]	4	Output	Channel out port

When HASCSS = 0 where there can only be one processor, the Cross Trigger Channel interfaces of the processor are provided as expansion interfaces, CPU0CTICHIN[3:0] and CPU0CTICHOUT[3:0]. For more information of these interfaces, see *Arm® Cortex®-M55 Processor Technical Reference Manual* and *Arm® Cortex®-M85 Processor Technical Reference Manual*.

- For Cortex-M55 , these interfaces are synchronous to CPUCPU0CLK, and reside in the PD_CPU0 power domain and in the nCOLDRESETCPU0 reset domain.
- For Cortex-M85, these interfaces are synchronous to DEBUGCPU0CLK, and reside in the PD_DEBUG power domain and in the nCOLDRESETDEBUGCPU0 reset domain.

One or more of these interfaces must exist when DEBUGLEVEL > 0. Otherwise, this interface might not exist. If it does exist, it is tied or unused.

6.14.4 Cross Trigger Interface

When DEBUGLEVEL > 0 and HASCSS = 1, the CRSAS Ma2 includes a shared Cross Trigger Interface (CTI) module. Some trigger signals to and from the CTI are used within the system, while a number is made available to system expansion. The following table lists the trigger signals that are available for system expansion.

In the following table, NUM_EVENT_SLAVES and NUM_EVENT_MASTERS are defined by the NUM_EVENT_SLAVES and NUM_EVENT_MASTERS configuration of the css600_cti device respectively. For the CRSAS Ma2, both NUM_TRIG NUM_EVENT_SLAVES and

NUM_EVENT_MASTERS must be >= 8. For more information on the CTI, see *Arm® CoreSight™ System-on-Chip SoC-600M Technical Reference Manual*.

Table 6-7: CTI triggers

Signal name	Width	Direction	Description
CTIEVENTIN[<NUM_EVENT_SLAVES-1>:4]	4	Input	CTI Trigger inputs
CTIEVENTOUT[<NUM_EVENT_MASTERS-1>:6]	2	Output	CTI Trigger outputs

This interface is synchronous to DEBUGDEBUGCLK, resides in the PD_DEBUG power domain and nCOLDRESETDEBUG reset domain.

This interface must exist when DEBUGLEVEL > 0 and HASCSS = 1. Otherwise, this interface might not exist and if it does, it is tied or unused.

6.14.5 Debug APB Expansion interface

When DEBUGLEVEL > 0 and HASCSS = 1, the CRSAS Ma2 provides a Debug APB expansion interface so that partners can add more debug functionality to the Debug System. This interface is only accessible through:

- The debug interface using an external DAP when SECDBGSTAT.SYSDSSACCEN<n>_STATUS = 1.
- The system interconnect for CPU<n> when SECDBGSTAT.SYSDSSACCEN<n>_STATUS = 1, or for **IMPLEMENTATION DEFINED** manager on expansion interfaces when SECDBGSTAT.SYSDSSACCENX_STATUS = 1.

For more information of the address mapping of this interface, see HASCSS = 1.

This interface is synchronous to DEBUGDEBUGCLK, is in the nCOLDRESETDEBUG reset domain and resides in the PD_DEBUG power domain.

Table 6-8: Debug APB Expansion interface

Signal name	Width	Direction	Description
DEBUGPRDATA[31:0]	32	Input	APB read data. Drives this bus during read cycles
DEBUGPREADY	1	Input	APB ready. Uses this signal to extend an APB transfer.
DEBUGPSLVERR	1	Input	Indicates a transfer failure. The APB peripherals are not required to support the PSLVERR pin.
DEBUGPADDR[31:2]	30	Output	The APB address bus for manager interface
DEBUGPSEL	1	Output	APB select. Indicates that the subordinate device is selected, and a data transfer is required.
DEBUGPENABLE	1	Output	APB enable. Indicates the second and subsequent cycles of an APB transfer.
DEBUGPWRITE	1	Output	APB RW transfer. Indicates an APB write access when HIGH, and an APB read access when LOW.
DEBUGPWDATA[31:0]	32	Output	Write data.

When DEBUGLEVEL = 0 or HASCSS = 0, this interface might not exist. If it exists, it is tied or unused.

6.14.6 CPU<n> External Peripheral interface EPPB

Each CPU<*n*> in the system provides an interface that allows users to add peripherals to the External PPB region that are private to each processor.

This is either one 32-bit AMBA 4 APB interface for a Cortex-M55, or two 32-bit AMBA 4 APB interfaces for a Cortex-M85. These are typically used for integration with additional CoreSight debug and trace components if necessary. Only data accesses are allowed on each of these interfaces privately from each processor at address `0xE0004_0000` to `0xE00F_FFFF`. Some of these regions are already used by peripherals like EWIC or reserved and others are available for integration of additional components.

Cortex-M55

If <*n*>TYPE is Cortex-M55, each interface that is associated to CPU<*n*> resides in the PD_CPU<*n*> power domain, the CPUCPU<n>CLK clock domain and the nCOLDRESETCPU<n> reset domain.

Cortex-M85

If <*n*>TYPE is Cortex-M85, each interface that is associated with CPU<*n*> core PPB resides in the PD_CPU<*n*> power domain, the CPUCPU<n>CLK clock domain and the nCOLDRESETCPU<n> reset domain. Each interface that is associated with CPU<*n*> Debug PPB resides in the PD_DEBUG power domain, the DEBUGCPU<n>CLK clock domain and the nCOLDRESETDEBUGCPU<n> reset domain.

For more information and for the address mapping, see CPU Private Peripheral Bus region, *Arm® Cortex®-M55 Processor Technical Reference Manual* or *Arm® Cortex®-M85 Processor Technical Reference Manual*.

6.14.7 ATB Trace interfaces

When DEBUGLEVEL = 2, the CRSAS Ma2 provides interfaces to output trace data to an expansion Trace Port Interface Unit (TPIU). The number and types of interfaces vary depending on the HASCSS configuration as follows:

- When HASCSS = 0 and there can only be one processor, the processor provides its ITM ATB Trace interface and an ETM ATB Trace interface directly for expansion. The processor's Trace synchronization and Trigger Interface are also provided for expansion. For more information on these interfaces, see *Arm® Cortex®-M55 Processor Technical Reference Manual* or *Arm® Cortex®-M85 Processor Technical Reference Manual*.

  All interfaces reside in the PD_DEBUG power domain and reside in the DEBUGCPU0CLK clock domain and nCOLDRESETDEBUGCPU0 reset domain.
- When HASCSS = 1, the CRSAS Ma2 provides a single ATB bus interface that is normally expected to connect to an external TPIU. This trace bus is synchronous to DEBUGDEBUGCLK, is reset using nCOLDRESETDEBUG and resides in the PD_DEBUG power domain.

Table 6-9: ATB Trace interface

Signal name	Width	Direction	Description
ATVALID	1	Output	A transfer is valid during this cycle. If LOW, all the other ATB signals must be ignored in this cycle.
ATID[6:0]	7	Output	An ID that uniquely identifies the source of the trace.
ATBYTE	IMPLEMENTATION DEFINED	Output	The number of bytes on ATDATA to be captured, minus 1. Note that is ATDATA is only eight bits wide, this signal may not exist.
ATDATA	IMPLEMENTATION DEFINED	Output	Trace data bus.
ATREADY	1	Input	Subordinate is ready to accept data.
AFVALID	1	Input	This is the flush signal. All buffers must be flushed because trace capture is about to stop.
AFREADY	1	Output	This is a flush acknowledge. Asserted when buffers are flushed.
SYNCREQ	1	Input	Trace synchronization request trace sinks.

When DEBUGLEVEL < 2 these interfaces might not exist. If they do exist, they are tied or unused.

6.14.8 Debug Authentication interface

The debug authentication signals of the subsystem reside in the PD_MGMT power domain, when PILEVEL = 2 its states are saved and restored when entering and then leaving the HIBERNATION1 lower power state, respectively. The state retention is **IMPLEMENTATION DEFINED** and may be performed using shadow registers in PD_AON. Throughout HIBERNATION1 low power state these signals must be driven to PD_AON if used in PD_AON and remain static.

When configuration option LCM_KMU_SAM_PRESENT = 0

- The input signals in the following table define the debug authentication signal values, when they are not overridden by the internal Secure debug configuration registers.

When configuration option LCM_KMU_SAM_PRESENT = 1

- The following input signals absent from the interface and controlled by the LCM Debug Control Unit and the LCM_DCU_FORCE_DISABLE register

Table 6-10: Debug authentication interface inputs

Signal name	Width	Direction	Description
DBGENIN	1	Input	Debug Enable input
NIDENIN	1	Input	Non-Invasive Debug Enable input
SPIDENIN	1	Input	Secure Privileged Invasive Debug Enable input
SPNIDENIN	1	Input	Secure Privileged Non-Invasive Debug Enable input
DAPACCENIN	1	Input	External Debug Access Enable input
DAPDSSACCENIN	1	Input	DAP to Debug System Access Enable input
SYSDSSACCEN<n>IN	1	Input	CPU to Debug System through System Interconnect Access Enable Input. This signal must exist when HASCSS = 1. There is one bit per CPU<n> to indicate which processor is allowed access.
SYSDSSACCENXIN	1	Input	**IMPLEMENTATION DEFINED** Manager to Debug System through System Interconnect Access Enable Input. This signal must exist when HASCSS = 1. The Manager selected for access control is IMPLEMENTATION DEFINED

The merged debug authentication signals are made available as outputs to the rest of the system. These are defined as the output signals in the following table.

For more information on how the output is generated, see Secure Debug Configuration Registers and LCM Debug Control Unit.

Table 6-11: Debug authentication interface outputs

Signal name	Width	Direction	Description
DBGEN	1	Output	Merged Debug Enable Output
NIDEN	1	Output	Merged Non-Invasive Debug Enable Output
SPIDEN	1	Output	Merged Secure Privileged Invasive Debug Enable Output
SPNIDEN	1	Output	Merged Secure Privileged Non-Invasive Debug Enable Output
DAPACCEN	1	Output	Merged External Debug Access Enable Output
DAPDSSACCEN	1	Output	Merged DAP to Debug System Access Enable Output.
SYSDSSACCENX	1	Output	Merged Defined Manager to Debug System through System Interconnect Access Enable Output. This signal must exist when HASCSS = 1. The Manager selected for access control is **IMPLEMENTATION DEFINED**.

6.15 Lifecycle Manager Expansion interfaces

The following section details interfaces that must exist when LCM_KMU_SAM_PRESENT = 1.

6.15.1 Lifecycle Manager Lifecycle Indication interface

The Lifecycle Indication interface indicates the lifecycle state of the system through different stages of manufacture and deployment of the final product.

The life-cycle signals indicate the following information:

- Current life cycle including the test or production mode
- Validity of the life cycle
- Fatal error state

All signals are synchronous to MGMTSYSCLK, reset on the nCOLDRESETMGMT and are on the PD_MGMT power domain. The context of Lifecycle Manager must be saved and restored (retained) over HIBERNATION0. All signals that are driving PD_AON logic have to be latched and hold during HIBERNATION0.

The following table shows the output signals on the Lifecycle Indication interface.

Table 6-12: Lifecycle Manager Lifecycle Indication interface

Signal name	Width	Direction	Description
LCS	3	Output	Lifecycle State values: • 3'b000 Chip Manufacture (CM) • 3'b001 Device Manufacture (DM) • 3'b101 Secure Enable (SE) • 3'b111 Return to Manufacturer (RMA) These are driven by the lcs output signal of the LCM.
LCSVALID	1	Output	Lifecycle State values on LCS is valid, where: • '0' LCS is not valid. • '1' LCS is valid. These are driven by the lcs_is_valid output signal of the LCM.
TPMODE	2	Output	Test or Production Chip Indication. It shows whether the LCM is in test or production mode Lifecycle State • 2'b00 Virgin • 2'b01 Test Chip Indication (TCI). • 2'b10 Production Chip Indication (PCI) • 2'b11 Invalid These are driven by the tp_mode output of the LCM.
FATALERR	1	Output	Indicates that the LCM has been disabled because of a critical fatal error caused internally by LCM FSM or set by the programmer. The fatal error indication takes precedence on all life-cycle states, meaning that after the fatal error is set, LCM is not functional anymore for the current power cycle. These are driven by the fatal_err output signal of the LCM.

6.15.2 Lifecycle Manager Debug Control Unit interface

The Lifecycle Manager Debug Control Unit interface provides signals that can be used to control the Debug Authentication interface signals and other test, debug, and security-related functionality in SoC.

See debug authentication interface for more information on the signals of the interface.

All signals are synchronous to MGMTSYSCLK, reset on the nCOLDRESETMGMT and are on the PD_MGMT power domain. The context of Lifecycle Manager, including the Debug Control Unit interface must be saved and restored (retained) over HIBERNATION0. All signals that are driving PD_AON logic have to be latched and held during HIBERNATION0.

The following table shows the output signals on the Lifecycle Manager Debug Control Unit interface.

Table 6-13: Lifecycle Manager Debug Control Unit interface

Signal name	Width	Direction	Description
DCUEN[127:32]	96	Output	Debug Control Enable Values. These are connected to dcu_en[127:32] output of the LCM. These are intended to connect to SoC debug controls or feature controls, for example, frequency control, licensed features.

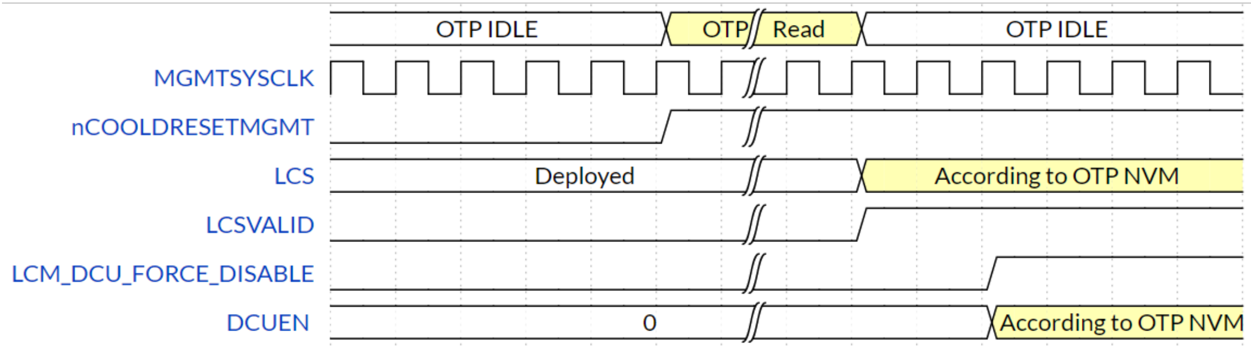
Note

dcu_en [31:0] of the LCM are reserved for Debug Authentication interface control and for internal usage, for example, BIST enable or ECOs. For more information, see LCM Debug Control Unit.

The following figure shows the DCUEN behaviour during Reset and following Reset deassertion regarding the Lifecycle State data obtained from the LCM NVM (OTP).

For more information on the behaviour of the LCM's dcu_en signals, see *Arm® Key Management Unit (KMU) Specification* and *Arm® Lifecycle Manager (LCM) Specification*.

Figure 6-2: DCUEN Behaviour during Reset and following Reset deassertion



6.15.3 Lifecycle Manager Non-Volatile Memory interface

The Lifecycle Manager Non-Volatile Memory interface is the Lifecycle Manager interface to the *One-Time Programmable* memory (OTP).

- NVM interface clock is synchronous to MGMTSYSCLK.
- NVM interface resides in the PD_MGMT power domain.
- NVM interface reset is nCOLDRESETMGMT.

The context of Lifecycle Manager must be saved and restored (retained) over HIBERNATION0. All signals that are driving PD_AON logic have to be latched and hold during HIBERNATION0.

For more information on the LCM NVM interface, refer to the OTP APB3 manager interface in *Arm® Lifecycle Manager (LCM) Specification*.

6.15.4 Lifecycle Manager LFSR seed interface

The Lifecycle Manager includes an interface to read a seed value that is used to generate masking and random delay random values. The seed is expected to be generated by a true random number generator from an entropy source.

- LFSR seed interface clock is synchronous to MGMTSYSCLK.
- LFSR seed interface resides in the PD_MGMT power domain.
- LFSR seed interface reset is nCOLDRESETMGMT.

The context of Lifecycle Manager must be saved and restored (retained) over HIBERNATION0.

Table 6-14: Lifecycle Manager LFSR seed interface

Signal name	Width	Direction	Description
LFSRDATA	64	Input	The LFSR seed data bits. This is connected to the lfsr_data input of the LCM
LFSRVALID	1	Input	The LFSR seed data bits are valid. This is connected to the lfsr_valid input of the LCM

For more information on the LFSR seed interface, refer to the LFSR seed signals in *Arm® Lifecycle Manager (LCM) Specification*.

6.16 Key Management Unit interface

The CRSAS Ma2 uses KMU for hardware-based Secure key management. This interface exits when LCM_KMU_SAM_PRESENT = 1.

The Key Management Unit interface is an AMBA AHB5 Manager interface that is used by the KMU to export the keys in KMU to the target crypto devices in the system applying a hardware-based method.

The system integrator can route the traffic of this interface privately or through the shared interconnect to all crypto devices.

- The KMU interface clock is synchronous to MGMTSYSCLK.
- The KMU interface resides in the PD_MGMT power domain.
- The KMU interface is reset on nCOLDRESETMGMT. However, during Secure Asset Provisioning Flow when LCMRSTREQ is asserted, the nWARMRESETMGMT also resets the KMU. For more information, see Reset infrastructure.

For more information regarding the KMU interface, see *Arm® Key Management Unit (KMU) Specification*.

6.17 Security Control Expansion signals

The the CRSAS Ma2 provides additional status and control signals to handle additional MSC, MPC, PPC, and Bridges with write buffers in the expansion system. These signals allow all the components to be controlled using the same set of security control registers already implemented within the subsystem.

All signals in this section are synchronous to SYSSYSCLK, and the SYSSYSCLK Q-Channel Device Interface is needed to control the availability of SYSSYSCLK. These signals reside in the PD_SYS power domain and in the nWARMRESETSYS reset domain.

Note

While the CRSAS Ma2 defines a full set of signals in this document, many of these interfaces and some of their individual bits can be unimplemented or disabled. These are defined as configuration options in Configuration options. How each configuration option is implemented is **IMPLEMENTATION DEFINED**.

6.17.1 Memory Protection Controller Expansion

The CRSAS Ma2 supports up to 16 MPCs to be added to the expansion system.

The SMPCEXPSTATUS signal allows the interrupts of the MPCs to be internally merged to the single MPC Combined interrupt.

Table 6-15: MPC Expansion Interrupt Status input

Signal name	Width	Direction	Description
SMPCEXPSTATUS	16	Input	Interrupt Status inputs from all Expansion Memory Protection Controllers. These are programmed through the SECMPCINTSTAT.SMPCEXP_STATUS register fields in the Secure Access Configuration Register block and are used to raise an interrupt using the MPC Combined Interrupt. Individual bits of this interface can be unimplemented or disabled, which results in the associated register bit field being **RAZ/WI**.

6.17.2 Peripheral Interconnect Peripheral Protection Controller Expansion

The CRSAS Ma2 supports up to four additional PPCs to be added to the Peripheral Interconnect in the expansion system.

The following signals are provided to control the PPC <i> where i is {0-3}.

Table 6-16: Peripheral Interconnect PPC Expansion interface

Signal Name	Width	Direction	Description
SPERIPHPPCEXPSTATUS	4	Input	Peripheral Interconnect PPC Interrupt Status Input. Each bit SPERIPHPPCEXPSTATUS[i] is to be connected to a single PPC <i>. The SPERIPHPPCEXPSTATUS[i] bit is associated to the SECPPCINTSTAT.SPERIPHPPCEXP_STATUS[i] register field. Individual bits of this interface can be unimplemented or disabled, which results in the associated register bit field being **RAZ/WI**.

Signal Name	Width	Direction	Description
SPERIPHPPCEXPCLEAR	4	Output	Peripheral Interconnect PPC Interrupt Clear Output. Each bit SPERIPHPPCEXPCLEAR[i] is to be connected to a single PPC <i>. The SPERIPHPPCEXPCLEAR[i] bit is associated to the SECPPCINTCLR.SPERIPHPPCEXP_CLR[i] register field. Individual bits of this interface can be unimplemented or disabled which results in the associated register bit field being **RAZ/WI**.
PERIPHNSPPCEXP<i>	16	Output	Peripheral Interconnect PPC Non-secure Gating Control. These are a set of multiple bit interfaces, and each interface connects to PPC <i>. When each bit 'j' of an interface is HIGH, it defines a specific <j> interface that the target PPC controls as Non-secure access only. Each bit PERIPHNSPPCEXP<i>[j] is driven by the PERIPHNSPPCEXP<i>[j] register. Individual bits of this interface can be unimplemented or disabled, which results in the associated register bit field being **RAZ/WI**.
PERIPHPPPCEXP<i>	16	Output	Peripheral Interconnect PPC Privilege Gating Control. These are a set of multiple bit interfaces and each interface connects to PPC <i>. When each bit PERIPHPPPCEXP<i>[j] of an interface is HIGH, it defines the <j> interface that the target PPC <i> controls as both privileged and unprivileged access. Else, it is privileged access only. Each bit PERIPHPPPCEXP<i>[j] is selected from either PERIPHSPPPCEXP<i>[j] if PERIPHNSPPCEXP<i>[j] is '0' or PERIPHNSPPPCEXP<i>[j] otherwise. Individual bits of this interface can be unimplemented or disabled, which results in the associated register bit fields that contributes to this control signal being **RAZ/WI**.

6.17.3 Main Interconnect Peripheral Protection Controller Expansion

The CRSAS Ma2 supports up to four additional PPCs to be added to the Main Interconnect in the expansion system.

The following signals are provided to control each PPC<i> where i is {0-3}.

Table 6-17: Main Interconnect PPC Expansion interface

Signal name	Width	Direction	Description
SMAINPPCEXPSTATUS	4	Input	Main Interconnect PPC Interrupt Status Input. Each bit SMAINPPCEXPSTATUS[i] is to be connected to a single PPC<i> The SMAINPPCEXPSTATUS[i] bit is associated to the SECPPCINTSTAT.SMAINPPCEXP_STATUS[i] register field. Individual bits of this interface can be unimplemented or disabled, which results in the associated register bit field being **RAZ/WI**.

Signal name	Width	Direction	Description
SMAINPPCEXPCLEAR	4	Output	Main Interconnect PPC Interrupt Clear Output. Each bit SMAINPPCEXPCLEAR[i] is to be connected to a single PPC<i>. The SMAINPPCEXPCLEAR[i] bit is associated to the SECPPCINTCLR.SMAINPPCEXP_CLR[i] register field. Individual bits of this interface can be unimplemented or disabled, which results in the associated register bit field being **RAZ/WI**.
MAINNSPPCEXP0<i>	16	Output	Main Interconnect PPC Non-secure Gating Control. These are a set of multiple bit interfaces, and each interface connects to PPC<i>. When each bit MAINNSPPCEXP<i>[j] of an interface is HIGH, it defines the <j> interface that the target PPC<i> controls as Non-secure access only. Each bit MAINNSPPCEXP<i>[j] is driven by the MAINNSPPCEXP<i>[j] register. Individual bits of this interface can be unimplemented or disabled which results in the associated register bit field being **RAZ/WI**.
MAINPPPCEXP<i>	16	Output	Main Interconnect PPC Privilege Gating Control. These are a set of multiple interfaces and each interface connects to PPC<i>. When each bit MAINPPPCEXP<i>[j] of an interface is HIGH it defines the <j> interface that the target PPC<i> controls as both privileged and unprivileged access. Else, it is privileged access only. Each bit MAINPPPCEXP<i>[j] is selected from either MAINSPPPCEXP<i>[j] if MAINNSPPCEXP<i>[j] is '0' or MAINNSPPPCEXP<i>[j] otherwise. Individual bits of this interface can be unimplemented or disabled which results in the associated register bit fields that contributes to this control signal being **RAZ/WI**.

6.17.4 Manager Security Controller Expansion

The CRSAS Ma2 supports up to 16 additional Manager Security Controllers (MSC) to be added to the expansion system.

The following signals are provided to control each MSC<i> where <i> is {0-15}:

Table 6-18: MSC Expansion interface

Signal name	Width	Direction	Description
SMSCEXPSTATUS	16	Input	MSC Interrupt Status Input. Each bit SMSCEXPSTATUS[i] is to be connected to a single MSC<i>. These are associated with the SECMSCINTSTAT.SMSCEXP_STATUS register field. Individual bits of this interface can be unimplemented or disabled, which results in the associated register bit field being **RAZ/WI**.
SMSCEXPCLEAR	16	Output	MSC Interrupt Clear Output. Each bit SMSCEXPCLEAR[i] is to be connected to a single MSC<i>. These are associated with the SECMSCINTCLR.SMSCEXP_CLR register field. Individual bits of this interface can be unimplemented or disabled, which results in the associated register bit field being **RAZ/WI**.

Signal name	Width	Direction	Description
NSMSCEXP	16	Output	MSC Non-secure Configuration. Each bit NSMSCEXP[i] is to be connected to a single MSC<i>. Set HIGH to configure a manager as Non-secure. These are associated with the NSMSCEXP.NS_MSCEXP register field. Individual bits of this interface can be unimplemented or disabled. Any disabled bit of this interface that still exist must be tied HIGH.

6.17.5 Bridge Buffer Error Expansion

The CRSAS Ma2 supports up to 16 additional bridges with buffer error signalling to be added to the expansion system.

Table 6-19: Bridge Error Interrupt Expansion interface

Signal name	Width	Direction	Description
BRGEXPSTATUS	16	Input	Bridge Error Interrupt Status Input. Each bit BRGEXPSTATUS[i] is to be connected to a single bridge <i> where *i* is 0 to 15. These are associated with the BRGINTSTAT.BRGEXP_STATUS register field. Individual bits of this interface can be unimplemented or disabled, which results in the associated register bit field being **RAZ/WI**.
BRGEXPCLEAR	16	Output	Bridge Error Interrupt Clear Output. Each bit BRGEXPCLEAR[i] is to be connected to a single bridge <i> where *i* is 0 to 15. These are associated with the BRGINTCLR.BRGEXP_CLR register field. Individual bits of this interface can be unimplemented or disabled, which results in the associated register bit field being **RAZ/WI**.

6.17.6 Other Security Expansion signals

The CRSAS Ma2 supports other security expansion signals that are needed by PPCs and MSCs in the expansion system.

The following table lists signals related to security that is needed by PPCs and MSCs in the expansion system.

Table 6-20: Other Security Expansion signals

Signal name	Width	Direction	Description
SECRESPCFG	1	Output	This output configures how to respond to an access when a security violation occurs. • 0: RAZ/WI • 1: Bus error This output is controlled by the SECRESPCFG register.

Signal name	Width	Direction	Description
ACCWAITn	1	Output	This request output is used to control any external gating unit that may be required to block accesses to the system through the Main and Peripheral Interconnect Expansion interfaces. • 1: No gating • 0: Access gated This output is controlled by the BUSWAIT register.
ACCWAITNSTATUS	1	Input	This status output is used to indicate the current state of any external gating unit that may be used to block access to the system through the Main and Peripheral Interconnect Expansion interfaces • 1: No gating • 0: Access gated This input can be read using the BUSWAIT register.

6.18 Sensor alarm interface

The Sensor alarm interface provides expansion interfaces to the Security Alarm Manager (SAM).

See SAM for more information about the manager.

The Sensor Alarm interface allows:

- Additional security
- Fault injection countermeasures to be added to the system to raise an alarm
- Trigger to respond to events

Inputs

SAMEVENT[15:0]

External sensor event that is provided as optional expansion to the SAM. These are connected to SAM events 48..63 respectively. See Security Alarm Manager incoming events allocation.

SAMSENSORREADY

External Sensors Ready signal that is provided as expansion to the SAM. This signal must be set when all the external sensor events are stable. If no external event is connected, the External Sensors Ready signal must be constantly set.

Outputs

SAMSTATUS[63:0]

These are the SAM captured events. The indications reflect the matching SAMES0 and SAMES1 registers. See Security Alarm Manager incoming events allocation.

SAMACTION[7:0]

SAM Response action. Used to trigger resets or interrupts. Each signal indicates, when set, that its respective sensor reported an alarm condition. Some of the actions are used internally in the system while also presented on the interface. The SAM response settings are detailed in Security Alarm Manager Response Action Settings.

SAMCONFIGDONE

SAM configuration has been finished. It indicates that the SAM is configured properly, sensors are ready so the software boot flow may proceed.

All SAM signals are synchronous to AONCLK and are on the PD_AON power domain if not defined otherwise. SAMSTATUS[63:0] is reset on the nPORESET signal and the rest are reset on the nCOLDRESETAON signal.

6.19 Clock configuration interface

The CRSAS Ma2 provides a set of control and status signals for some of the clocks to configure generators or dividers that might exist in the expansion system.

Each set of control and status signal and even individual bits of this interface can be unimplemented or disabled. This, and how they are used are **IMPLEMENTATION DEFINED**, but if a set exists, then the default value of each signal at reset must be set to a value to allow the input clock to run at a default clock rate to allow the system to at least boot without software configurating these sets of registers. The reset value can be configured.

All signals in this interface reside in the PD_AON power domain, are synchronous to AONCLK and are on the nCOLDRESETAON reset domain.

Table 6-21: Clock configuration interface signals

Signal name	Width	Direction	Description
CPU<n>CLKCFG	4	Output	These control outputs provide a set of four-bit outputs that allows the system to configure an external clock generation logic that drives CPU<n>CLK. These output signals are driven by the register fields CLK_CFG0.CPU<*n*>CLKCFG. Any unimplemented bits result in the associated register bit field being **RAZ/WI**.
CPU<n>CLKCFGSTATUS	4	Input	These sets of four-bit input signals are used to read the status of any external clock generation logic that drives CPU<n>CLK. The values on this interface can be read by the register fields CLK_CFG0.CPU<*n*>CLKCFGSTATUS. Any unimplemented bits result in the associated register bit field being **RAZ/WI**.
SYSCLKCFG	4	Output	This control output provides four-bit outputs that allows the system to configure an external clock generation logic that drives SYSCLK. This output is driven by the register fields CLK_CFG1.SYSCLKCFG. Any unimplemented bits result in the associated register bit field being **RAZ/WI**.
SYSCLKCFGSTATUS	4	Input	This four-bit input signal is used to read the status of any external clock generation logic that drives SYSCLK. The values on this interface can be read by the register fields CLK_CFG1.SYSCLKCFGSTATUS. Any unimplemented bits result in the associated register bit field being **RAZ/WI**.

Signal name	Width	Direction	Description
AONCLKCFG	4	Output	This control output provides a four-bit output that allows the system to configure an external clock generation logic that drives AONCLK. This output is driven by the register fields CLK_CFG1.AONCLKCFG. Any unimplemented bits result in the associated register bit field being **RAZ/WI**.
AONCLKCFGSTATUS	4	Input	This four-bit status output is used to read the status of any external clock generation logic that drives AONCLK. The values on this interface can be read by the register fields CLK_CFG1.AONCLKCFGSTATUS. Any unimplemented bits result in the associated register bit field being **RAZ/WI**.
NPU<m>CLKCFG	4	Output	These control outputs provide a set of four-bit outputs that allows the system to configure an external clock generation logic that drives NPU<m>CLK. These outputs are driven by the register fields CLK_CFG2.NPU<*m*>CLKCFG. Any unimplemented bits result in the associated register bit field being **RAZ/WI**.
NPU<m>CLKCFGSTATUS	4	Input	These sets of four-bit status inputs are used to read the status of any external clock generation logic that drives NPU<m>CLK. The values on this interface can be read by the register fields CLK_CFG2.NPU<*m*>CLKCFGSTATUS. Any unimplemented bits result in the associated register bit field being **RAZ/WI**.

6.20 Miscellaneous signals

Additional miscellaneous signals are available for the CRSAS Ma2.

The following table lists the names, widt, and direction of the other miscellaneous top-level signals. It also gives information on what the signals are synched to and what power domains they belong to.

Table 6-22: Other miscellaneous top-level signals

Signal name	Width	Direction	Sync to	Power domain	Description
LOCKNSVTOR<n>	1	Input	CPU<n>CLK	PD_CPU <n>	Disables writes to the CPU<n> VTOR_NS register. When HIGH, this input prevents changes to the Non-secure vector table base address register of CPU<n>. If not used, tie to LOW, tying this signal HIGH causes loss of vector table control.

Signal name	Width	Direction	Sync to	Power domain	Description
LOCKNSMPU<n>	1	Input	CPU<n>CLK	PD_CPU <n>	This input disables writes to CPU<n> registers that are associated with the Secure Memory Protection Unit (MPU) region from software or from a debug agent connected to the processor. • MPU_CTRL • MPU_RNR • MPU_RBAR • MPU_RLAR • MPU_RBAR_An • MPU_RLAR_An When HIGH, this input prevents changes to the memory regions which have been programmed in the Secure MPU. All writes to these registers are ignored. If not used, tie to LOW, tying this signal HIGH causes loss of Secure Memory Protection Unit (MPU) control.
CPU<n>WAITCLR	1	Input	SYSCLK	PD_MGMT	When HIGH, clears the register fields CPUWAIT.CPU<n>WAIT. This allows an external entity to release a processor that is already waited by CPUWAIT to start execution. When set to '1', this signal must be held at '1' until CPU<n>WAITCLRRESP is '1'. While this signal is clocked using SYSCLK, this input can optionally be implemented as an asynchronous input.
CPU<n>WAITCLRRESP	1	Output	SYSCLK	PD_MGMT	This signal provides a response to the CPU<n>WAITCLR request. When CPU<n>WAITCLR is '1' and CPUWAIT.CPU<n>WAIT is '0', this signal is set to '1' until CPU<n>WAITCLR request goes '0'.
NSWDRSTREQSTATUS	1	Output	SYSCLK	PD_AON	Non-secure watchdog reset request status. This output is '1' when the Non-secure Watchdog is raising a reset request and RESET_MASK.NSWDRSTREQEN is '1'. When set to HIGH, it does not return to low unless a reset occurs clearing this status. This output is optional, but must exist when COLDRESET_MODE = 1.
SWDRSTREQSTATUS	1	Output	SYSCLK	PD_AON	Secure watchdog reset request status. This output is '1' when the Secure Watchdog is raising a reset request. When set to HIGH, it does not return to low unless a reset occurs clearing this status. This output is optional, but must exist when COLDRESET_MODE = 1.
SSWDRSTREQSTATUS	1	Output	SYSCLK	PD_AON	Secure Privileged SLOWCLK watchdog reset request status. This output is '1' when the SLOWCLK Watchdog is raising a reset request. When set to HIGH, it does not return to low unless a reset occurs clearing this status. This output is optional, but must exist when COLDRESET_MODE = 1.

Signal name	Width	Direction	Sync to	Power domain	Description
SAMCRSTREQSTATUS	1	Output	AONCLK	PD_AON	Security Alarm Manager (SAM) cold reset request. This reset request signal is assigned by Security Alarm Manager Response Action Settings. When this signal is asserted by SAM, a system wide cold reset is performed. Since the parts of the SAM and hence the response action outputs are not on cold reset, actions may remain set during cold reset. Hence the system is sensitive only to its positive edge. This signal must exist when LCM_KMU_SAM_PRESENT = 1 and COLDRESET_MODE = 1. If LCM_KMU_SAM_PRESENT = 0, it is equivalent to SAMCRSTREQ being tied LOW.
RESETREQSTATUS	1	Output	SYSCLK	PD_AON	Hardware Reset Request status. This output is set to '1' if RESETREQ input is '1'. When set to HIGH, it must not be cleared unless the system is reset. This output is optional, but must exist when COLDRESET_MODE = 1.
SWCOLDRSTREQSTATUS	1	Output	SYSCLK	PD_AON	Software Reset Request Status. This output is '1' when SWRESET.SWCOLDRESETREQ is set to '1'. When set to HIGH, it must not be cleared unless the system is reset while restores the register field to '0'. This output is optional, but must exist when COLDRESET_MODE = 1.
CPU<n>LOCKUP	1	Output	AONCLK	PD_AON	Processor Lockup Status. There is one bit per processor. Each bit indicates if the associated CPU<n> has lockup. This signal is an output directly from CPU<n>.
CPU<n>HALTED	1	Output	CPU<n>CLK	PD_CPU <n>	Processor Halted Status. There is one bit per processor. Each bit indicates if the associated CPU<n> has halted. This signal is an output directly from CPU<n>.
CPU<n>EDBGRQ	1	Input	CPU<n>CLK	PD_AON, PD_CPU <n>	External request for CPU<n> to enter halt mode. This signal is an input directly to CPU<n> and also to EWIC<n>.
CPU<n>DBGRESTART	1	Input	CPU<n>CLK	PD_CPU <n>	Request for CPU<n> to perform synchronised exit from halt mode. Forms a handshake with CPU<n>DBGRESTARTED. This signal is an input directly to CPU<n>.
CPU<n>DBGRESTARTED	1	Output	CPU<n>CLK	PD_CPU <n>	Acknowledges CPU<n>DBGRESTART. This signal is an output directly from CPU<n>.

For more information on the CPU<n> registers, see *Arm®v8-M Architecture Reference Manual*.

7. Configuration options

The CRSAS Ma2 specification is configurable, which allow systems based on this specification to scale across the performance, power, and area requirement of the market.

An implementation of the CRSAS Ma2 can support a subset of the configuration options defined here. An implementation of the subsystem, however, cannot support additional configuration options unless they are limited to micro-architectural features and are features that are orthogonal to existing configuration. Because of this, an implementation cannot add additional legal values to the existing configuration options. The method by which the configurations are supported is **IMPLEMENTATION DEFINED**.

7.1 Main system configuration options

The following table lists the main system configuration options for the CRSAS Ma2.

Table 7-1: The CRSAS Ma2 main system configurations

Configuration option name	Legal values	Description
NUMCPU	0-3	The number of Cortex-M processor cores in the subsystem. The number of cores is equal to NUMCPU + 1. When PILEVEL = 0, NUMCPU must be set to '0'.
CPU<n>TYPE	0, 3, 4	The type of processor that is integrated: • 0: Not implemented • 3: Cortex-M55 • 4: Cortex-M85 • Others: Reserved CPU0TYPE must not be '0' and sparse processors are not supported. Therefore CPU0TYPE to CPU<NUMCPU>TYPE must not be '0's.
NUMNPU	0-4	The number of Ethos NPU cores in the subsystem. The number of cores is equal to NUMNPU.
NPU<m>TYPE	0-2	The type of NPU that is integrated, if any: • 0: Not implemented • 1: Ethos-U55 • 2: Ethos-U65 • Others: Reserved This configuration option must exist when NUMNPU != 0. Sparse NPUs are not supported. Therefore, NPU0TYPE must not be '0's.

Configuration option name	Legal values	Description
NPU<m>PORSLRST	0-1	The default security level that each NPU resets to. • 0: Secure State • 1: Non-secure State • Others: Reserved This configuration option must exist when NUMNPU != 0.
NPU<m>PORPLRST	0-1	The default privilege level that each NPU resets to. • 0: Unprivileged State • 1: Privileged State • Others: Reserved This configuration option only exists when NUMNPU != 0. We recommend that this is set to 1.
NUMDMA	0-1	The number of DMA cores present in the system.
DMATYPE	0, 1	The type of DMA that is integrated: • 0: Not implemented • 1: DMA-350 • Others: Reserved This configuration option must exist when NUMDMA != 0.
PILEVEL	0-2	Power Infrastructure Level. Defines the implemented power structure of the system: • 0: Basic Power Structure • 1: Intermediate Power Structure • 2: Advanced Power infrastructure • Others: Reserved
ROMADDRWIDTH	0,14-24	Defines the existence and the address width for the Boot ROM as $2^{\text{ROMADDRWIDTH}}$ bytes. When set to zero, the ROM does not exist.
NUMVMBANK	0-4	Selects the number of Volatile Memory Banks.
VMADDRWIDTH	14 to (24- ceil($\log_2$ (NUMVMBANK))	Defines the address width for all Volatile Memory Banks when NUMVMBANK > 0. This then defines the size of each bank as $2^{\text{VMADDRWIDTH}}$ bytes.
VMMPCBLKSIZE	3-15	Defines the block size of the MPC associated with all Volatile Memory Banks. Volatile Memory block size = $2^{(\text{VMMPCBLKSIZE} + 5)}$ bytes.
LCM_KMU_SAM_PRESENT	0-1	It defines whether Hardware Lifecycle Management, Key Management Unit and Security Alarm Manager is included to meet PSA level 2 requirements. • 0: No • 1: Yes
LCM_DCU_FORCE_DISABLE_INIT	bit[i] = not(bit[i+1]) where i is {0-21} bit[j] = {0,1} where j is {22-31}	Defines initial value of LCM_DCU_FORCE_DISABLE register. Recommended value is `0xFFD5_5555` See the bits allocation and more details in LCM Debug Control Unit. When LCM_KMU_SAM_PRESENT = 0 this value is not used.

Configuration option name	Legal values	Description
HASCSS	0-1	Defines whether the CoreSight SoC-600M based Debug infrastructure is included. • 0: No • 1: Yes HASCSS must be to 1 when NUMCPU > 0.
HASCPU<n>IWIC	0-1	Defines whether each processor has IWIC. • 0: No • 1: Yes
PDCMQCHWIDTH	0-4	Selects the width of Power Dependency Control Matrix Q-Channel interface that the system supports. When set to '0', the Power Dependency Control Matrix Q-Channel interface does not exist.
INITSVTOR<n>RST[31:7]	Any address values that resides in Secure world and is executable.	The value of CPU <n> Secure Vector table offset address register in the System Control Register. When ROMADDRWIDTH > 0 and the Boot ROM is present, the value of INITSVTOR<n>RST[31:7] is expected point to the ROM base address. When the ROMADDRWIDTH = 0 and the Boot ROM is not present, it is expected to point to an immutable, Secure, non-volatile storage that contains the first stage boot loader (FSBL).
DBGENSELDIS	When LCM_KMU_SAM_PRESENT = 1, the only legal value is 1. Else, both 0 and 1 are legal.	The DBGEN Selector Disable. • 0: No • 1: Yes, force DBGEN to use DBGENIN
NIDENSELDIS	When LCM_KMU_SAM_PRESENT = 1, the only legal value is 1. Else, both 0 and 1 are legal.	The NIDEN Selector Disable. • 0: No • 1: Yes, force NIDEN to use NIDENIN
SPIDENSELDIS	When LCM_KMU_SAM_PRESENT = 1, the only legal value is 1. Else, both 0 and 1 are legal.	The SPIDEN Selector Disable. • 0: No • 1: Yes. force SPIDEN to use SPIDENIN
SPNIDENSELDIS	When LCM_KMU_SAM_PRESENT = 1, the only legal value is 1. Else, both 0 and 1 are legal.	The SPNIDEN Selector Disable. • 0: No • 1: Yes, force SPNIDEN to use SPNIDENIN
DAPACCENSELDIS	When LCM_KMU_SAM_PRESENT = 1, the only legal value is 1. Else, both 0 and 1 are legal.	The DAPACCEN Selector Disable. • 0: No • 1: Yes, force DAPACCEN to use DAPACCENIN

Configuration option name	Legal values	Description
DAPDSSACCENSELDIS	When LCM_KMU_SAM_PRESENT = 1, the only legal value is 1 only. Else, both 0 and 1 are legal.	The DAPDSSACCEN Selector Disable. • 0: No • 1: Yes, force DAPDSSACCEN to use DAPDSSACCENIN
SYSDSSACCENSELDIS	When LCM_KMU_SAM_PRESENT = 1, the only legal value is 1. Else, both 0 and 1 are legal.	The SYSDSSACCEN Selector Disable. • 0: No • 1: Yes, force SYSDSSACCEN<n> and SYSDSSACCENX to use SYSDSSACCEN<n>IN This configuration option must exist when HASCSS = 1.
NSMSCEXPRST[15:0]	0-1 for each bit	The reset value for NSMSCEXP.NS_MSCEXP[15:0]. This value defines the security world of each expansion MSC when the PD_SYS power domain is powered up or is reset. It defines up to 16 MSCs, where each bit is: • 0: Secure • 1: Non-secure
ACCWAITNRST	0-1	The reset value of Bus access wait at reset. This defines whether the system blocks access from expansion managers that implement access gating into the Main and Peripheral Interconnect when the PD_SYS power domain is powered up or is reset. • 0: Blocks access • 1: Allows access
CPU<n>EXPNUMIRQ	0-X	Specifies the number of expansion interrupts for each processor. 'X' means the maximum number of interrupts the processor can support minus 32.
CPU<n>EXPIRQDIS	One bit per expansion interrupt, with each set to '0' or '1'.	Specifies for each processor whether each expansion interrupt bit is implemented or disabled.
CPU<n>INTNMIENABLERST	0-1	Specifies the Warm reset value of NMI_ENABLE.CPU<n>_INTNMI_ENABLE. This determines whether the internally generated interrupt sources can raise the NMI interrupt on each processor: • 0: Internal NMI sources are masked from driving NMI • 1: Interrupt NMI sources can drive NMI
CPU<n>EXPNMIENABLERST	0-1	Specifies the Warm reset value of NMI_ENABLE.CPU<n>_EXPNMI_ENABLE. This determines whether the CPU<n>EXPNMI top level pin is able to raise an NMI interrupt on each processor: • 0: CPU<n>EXPNMI is masked from driving NMI. • 1: CPU<n>EXPNMI is allowed to drive NMI.

Configuration option name	Legal values	Description
CPU<n>WAITRST	0-1	The (Primary) wait of the processor at boot control register CPU<n>WAIT reset value. • 0: Boot normally • 1: Wait at boot We recommend that this is set to '0', unless there are other reasons in the system or SoC to initially stop a processor from booting post reset. For example, there may be another higher security entity in the SoC that wants access to the system before allowing the processors to boot.
CPUWAITRST_MODE	0-1	Specifies reset signal used to reset all CPU<n>WAIT register values. • 0: nCOLDRESETAON • 1: nWARMRESETAON We recommend that this is set to '0', unless there are other reasons in the system or SoC that needs CPU<n>WAIT to also return to its reset values on Warm Reset.
INITSVTORRST_MODE	0-1	Specifies reset signal used to reset all the INITSVTOR<n> registers. • 0: nCOLDRESETAON • 1: nWARMRESETAON We recommend that this is set to '1', unless there are other reasons in the system or SoC that needs INITSVTOR<n> to also return to its reset values on Cold Reset.
CPU<n>CPUIDRST	Defined by SoC integrator.	A unique identity value defined for each processor in the system. Defines the values read at each CPU <n> local's CPU<n>_IDENTITY.CPUID register. Legal values are within 0 to 15, inclusive.
LOCKDCAIC	0-1	Disables access to all processor's instruction cache direct cache access registers DCAICLR and DCAICRR. Asserting this signal prevents direct access to the instruction cache Tag or Data RAM content. This is required when using eXecutable Only Memory (XOM).
COLDRESET_MODE	0-1	Cold Reset Mode. It defines if the watchdog timeouts, SAM Cold reset request, RESETREQ signal, and writes to the SWRESET register can be used to trigger a system Cold reset and therefore drive nCOLDRESETAON: • 0: Watchdogs, RESETREQ signal, and the SWCOLDRESETREQ register value contributes to Cold Reset. • 1: Watchdogs, RESETREQ signal, and the SWCOLDRESETREQ register value does not contribute to Cold Reset. When set to '1', an entity outside the subsystem is expected to observe the following signals to decide when to drive HOSTRESETREQ: • NSWDRSTREQSTATUS • SWDRSTREQSTATUS • SSWDRSTREQSTATUS • RESETREQSTATUS • SWCOLDRSTREQSTATUS • SAMCRSTREQSTATUS

Configuration option name	Legal values	Description
DEBUGLEVEL	0-2	Selects the debug level of the subsystem: • 0: Debug System does not exist. There is no Debug Access and no Trace support. • 1: Debug system exists without Trace support. Debug Access Interface(s) exists, but Trace is not supported. • 2: Debug system exists with Trace support. Both Debug Access Interface(s) and Trace Interface exist.
MPCEXPDIS[15:0]	0-1 for each bit	Disables support for individual bits on the SMPCEXPSTATUS bus. If MPCEXPDIS[i] = 1'b1, then either SMPCEXPSTATUS[i] does not exist, or if it does exist, is not used.
MSCEXPDIS[15:0]	0-1 for each bit	Disables support for individual bits on the SMSCEXPSTATUS, SMSCEXPCLEAR and NSMSCEXP buses. If MSCEXPDIS[i] = 1'b1, then either SMSCEXPSTATUS[i], SMSCEXPCLEAR[i] and NSMSCEXP[i] does not exist, or if they do exist, SMSCEXPSTATUS[i] is not used, SMSCEXPCLEAR[i] are tied LOW and NSMSCEXP[i] are tied HIGH.
BRGEXPDIS[15:0]	0-1 for each bit	Disables support for individual bits on the BRGEXPSTATUS and BRGEXPCLEAR buses. If BRGEXPDIS[i] = 1'b1, then either BRGEXPSTATUS[i] and BRGEXPCLEAR[i] does not exist, or if they do exist, BRGEXPSTATUS[i] is not used and BRGEXPCLEAR[i] are tied LOW.
PERIPHPPCEXP{0-3}DIS[15:0]	0-1 for each bit	Disables support for individual bits on the PERIPHNSPPCEXP{0-3} and PERIPHPPPCEXP{0-3} buses. If PERIPHPPCEXP{0-3}DIS[i] = 1'b1, then either PERIPHNSPPCEXP{0-3}[i] and PERIPHPPPCEXP{0-3}[i] does not exist, or if they do exist, PERIPHNSPPCEXP{0-3}[i] and PERIPHPPPCEXP{0-3}[i] are not used and tied LOW.
MAINPPCEXP{0-3}DIS[15:0]	0-1 for each bit	Disables support for individual bits on the MAINNSPPCEXP{0-3} and MAINPPPCEXP{0-3} buses. If MAINPPCEXP{0-3}DIS[i] = 1'b1, then either MAINNSPPCEXP{0-3}[i] and MAINPPPCEXP{0-3}[i] does not exist, or if they do, MAINNSPPCEXP{0-3}[i] and MAINPPPCEXP{0-3}[i] are not used and tied LOW.
CPU<n>CLKCFGRST[3:0]	`0x0 - 0xF`	CLK_CFG0.CPU<n>CLKCFG reset value. CPU<n>CLKCFGRST defines the CPU<n>CLKCFG output expected to be used for clock divider or generation configuration of CPU<n>CLK.
SYSCLKCFGRST[3:0]	`0x0 - 0xF`	CLK_CFG1.SYSCLKCFG reset value. SYSCLKCFGRST defines the SYSCLKCFG output expected to be used for clock divider or generation configuration of SYSCLK.
AONCLKCFGRST[3:0]	`0x0 - 0xF`	CLK_CFG1.AONCLKCFG reset value. AONCLKCFGRST defines the AONCLKCFG output expected to be used for clock divider or generation configuration of AONCLK.
CPU<n>RSTREQENRST	0-1	CPU<n>RSTREQEN reset value. CPU<n>RSTREQENRST defines the reset value of RESET_MASK.CPU<n>RSTREQEN that is used to mask the system reset request signal, often with the signal name SYSRESETREQ, from CPU <n>. When set to '0' at reset, CPU<n> is not able to cause a system Warm reset when setting its local AIRCR.SYSRESETREQ control in its Application Interrupt and Reset Control Register (AIRCR). Setting this to '1' allows the reset to be requested. This control must be set to '0' when LCM_KMU_SAM_PRESENT = 1.

Configuration option name	Legal values	Description
HASCPU<n>CPIF	0-1	Specifies whether the coprocessor interface is included for CPU<n>: • 0: Coprocessor interface is not included • 1: Coprocessor interface is included
SOCIMPLID[11:0]	Defined by SoC integrator.	The SoC integrator JEP106 code.
SOCREV[3:0]	Defined by SoC integrator.	The SoC minor revision code.
SOCVAR[3:0]	Defined by SoC integrator.	The SoC variant or major revision code.
SOCPRTID[11:0]	Defined by SoC integrator.	The SoC product identity code.
IMPLID[11:0]	IMPLEMENTATION DEFINED	The subsystem implementer JEP106 code.
IMPLREV[3:0]	IMPLEMENTATION DEFINED	The subsystem minor revision code.
IMPLVAR[3:0]	IMPLEMENTATION DEFINED	The subsystem variant or major revision code.
IMPLPRTID[11:0]	IMPLEMENTATION DEFINED	The subsystem product identity code.
CPU<n>MCUROMADDR[31:12]	Defined by SoC integrator.	The address pointer to MCU ROM table private to each processor core. Only the 20 most significant bits are configurable. All lower address bits are zeros. This configuration point must exist when HASCSS = 1.
CPU<n>MCUROMVALID	Defined by SoC integrator.	The address pointer to MCU ROM table private to each processor core is valid. This configuration point must exist when HASCSS = 1.

7.2 Lifecycle Manager related configurations

When LCM_KMU_SAM_PRESENT = 1, the Lifecycle Manager (LCM) configuration options must exist. The DCUEN signals from the LCM have default configuration values per the current Lifecycle (LCS), per Secure provisioning (SP) state and per the LCM setup of either test chip (TCI) or production chip (PCI).

See LCM for more information about the manager.

The following table shows the recommended default values for these configurations, which take into account the Debug Authentication interface assignment.

The complete list of configuration options of LCM are defined in details in *Arm® Lifecycle Manager (LCM) Specification*. The remaining bits of the Configuration options that are not in the below table are **IMPLEMENTATION DEFINED**.

Table 7-2: Recommended LCM configurations for the CRSAS Ma2

Configuration option name	Recommended values	Description
DISABLE_DIRECT_KEY_APB_MASKING	1	This setting requires additional hardware outside LCM for SCA counter measures CRSAS Ma2 does not support SCA counter measures
TCI_DEFAULT_DCU_VAL_LCS_CM[21:0]	`0x15_5555`	All Debug Authentication interface enabled

Configuration option name	Recommended values	Description
TCI_DEFAULT_DCU_VAL_LCS_DM[21:0]	`0x15_5555`	All Debug Authentication interface enabled
TCI_DEFAULT_DCU_VAL_LCS_SE[21:0]	`0x15_5555`	All Debug Authentication interface enabled
TCI_DEFAULT_DCU_VAL_LCS_RMA[21:0]	`0x15_5555`	All Debug Authentication interface enabled
PCI_DEFAULT_DCU_VAL_LCS_CM[21:0]	`0x15_5555`	All Debug Authentication interface enabled
PCI_DEFAULT_DCU_VAL_LCS_DM[21:0]	`0x15_5555`	All Debug Authentication interface enabled
PCI_DEFAULT_DCU_VAL_LCS_SE[21:0]	`0x2A_AAAA`	All Debug Authentication interface enabled This is the most Secure configuration
PCI_DEFAULT_DCU_VAL_LCS_RMA[21:0]	`0x15_5555`	All Debug Authentication interface enabled
TCI_DCU_PERMANENT_DISABLE_MASK_VAL_LCS_CM[21:0]	`0xFF_FFFF`	All Debug Authentication interface can be enabled by software
TCI_DCU_PERMANENT_DISABLE_MASK_VAL_LCS_DM[21:0]	`0xFF_FFFF`	All Debug Authentication interface can be enabled by software
TCI_DCU_PERMANENT_DISABLE_MASK_VAL_LCS_SE[21:0]	`0xFF_FFFF`	All Debug Authentication interface can be enabled by software
TCI_DCU_PERMANENT_DISABLE_MASK_VAL_LCS_RMA[21:0]	`0xFF_FFFF`	All Debug Authentication interface can be enabled by software
PCI_DCU_PERMANENT_DISABLE_MASK_VAL_LCS_CM[21:0]	`0xFF_FFFF`	All Debug Authentication interface can be enabled by software
PCI_DCU_PERMANENT_DISABLE_MASK_VAL_LCS_DM[21:0]	`0xFF_FFFF`	All Debug Authentication interface can be enabled by software
PCI_DCU_PERMANENT_DISABLE_MASK_VAL_LCS_SE[21:0]	`0x2A_AAAA`	Software cannot enable any Debug Authentication interface, This is the most Secure configuration
TCI_DCU_PERMANENT_DISABLE_MASK_VAL_LCS_RMA[21:0]	`0xFF_FFFF`	All Debug Authentication interface can be enabled by software
DCU_SP_DISABLE_MASK_VAL[21:0]	`0x2A_AAAA`	Software cannot enable any Debug Authentication interface, This is the most Secure configuration

7.3 Key Management Unit related configurations

When LCM_KMU_SAM_PRESENT = 1, the Key Management Unit (KMU) configuration options must exist. The LCM and the CRSAS Ma2 related configuration options of KMU are defined in the foregoing table. This table shows the recommended default values for these configurations.

See Key Management Unit for more information about the unit.

These default values are set in accordance with the 7 hardware key slots that LCM maintains. It assumes that all these keys are of 256 bits. The complete list of configuration options of KMU are defined in details in *Arm® Key Management Unit (KMU) Specification*. The remaining bits of the Configuration options that are not in the below table are **IMPLEMENTATION DEFINED**. and have to be set according to the location and needs of the crypto devices in the system.

Table 7-3: Recommended Key Management Unit configurations for the CRSAS Ma2

Configuration option name	Recommended values	Description
KMUNKS	5	Selects the number of key slots that the KMU supports. This value is reflected in KMUBC.NKS. For more information, see *Arm® Key Management Unit (KMU) Specification*. KMUNKS must be greater than 2 to handle the 7 LCM HW keys.
KMUNHWKSLTS	7	Number of hardware key slots. The first KMUNHWKSLTS key slots are populated by the LCM and the rest are software key slots. This value is reflected in KMUBC.NHWKSLTS.
KMU_WMF_PRESENT	0	Specifies whether RWMF feature is enabled. 0 - forces WMD bits of all KMUKSC <i> registers to `0b1` and **RAO/WI**, overriding bit 21 of all KMUKSCRV<i> configuration options.
KMUKSCRV{0-6}	IMPLEMENTATION DEFINED	Reset value of the KMUKSC{0-6} register. This is a hardware key slot which holds 256 bits. Lock bits are set for hardware key slots by default. • LKS = `0b1` • LKSKR = `0b1` The remaining bits are **IMPLEMENTATION DEFINED**.
KMUDKPARV7	`0x0000_0000`	Reset value of the KMUDKPA7 register. This is the first software key slot.

7.4 Security Alarm Manager related configurations

When LCM_KMU_SAM_PRESENT = 1, the Security Alarm Manager (SAM) configuration options must exist.

See Security Alarm Manager for more information about the manager.

The CRSAS Ma2 related configuration options of the SAM are defined in the following table. The table shows the recommended default values for these configurations.

These default values are set in accordance with the default behaviour of SAM required before software setup. The complete list of configuration options of SAM are defined in details in *Arm® Security Alarm Manager (SAM) Specification*. The remaining bits of the configuration options that are not in the following table are **IMPLEMENTATION DEFINED** and have to be set according to the requirements of the system.

Table 7-4: Recommended Security Alarm Manager configurations for the CRSAS Ma2

Configuration option name	Recommended values	Description
SAMNRA	7	Number of SAM response actions. Selects the number of response outputs implemented in the SAM. For more information, see *Arm® Security Alarm Manager (SAM) Specification*. Minimum four response actions are required by the CRSAS Ma2.
SAMNEC	3	Number of SAM Event Counters. Selects the number of event counters implemented in the SAM. For more information, see *Arm® Security Alarm Manager (SAM) Specification*.
SAMRESETACTION[63:48] SAMRESETACTION[47:0]	**IMPLEMENTATION DEFINED** `0x0000_0000_0005`	The default behaviour out of reset for the Event Response Action Routing Enable of each event [63:0] respectively. The definition of each bit: • 0 – Do not route. Wait for SAM configuration. • 1 – Enable routing the latched event to its default response action which is defined in *Arm® Security Alarm Manager (SAM) Specification*. • Bit 0 is set to enable the default response for events from the SAM Configuration Integrity Checker errors • Bit 2 is set to enable the default response for LCM FATALERR from the Lifecycle Manager Lifecycle Indication interface and debug_signals_security_checker_err indication for DCUEN[21:0].

8. Programmers model

This section describes the functions and programmers model of the CRSAS Ma2.

8.1 System memory map overview

The high level system address map shows the high-level view of the memory map defined by the CRSAS Ma2. This memory map is divided into Secure and Non-secure regions. The memory alternates between Secure and Non-secure regions on 256Mbyte regions, with only a few address areas exempted from security mapping because they are related to debug functionality.

To provide memory blocks and peripherals that can be mapped either as Secure or Non-secure using the software, several address regions are aliased as shown in the table. The 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 Values column in the table specifies the security of each area along with its ID. The Non-secure Callable (NSC) settings of each region.

The following occur except when specifically stated:

- All accesses to unmapped regions of the memory result in bus-error response.
- When accessing unmapped address space within a mapped region taken by a peripheral, the access results in Read-As-Zero and Write-Ignored (RAZ/WI) except when specifically stated otherwise.
- Any accesses that result in security violations are either **RAZ/WI** or return a bus error response as defined by the SECRESPCFG register setting.

When an access results in bus-error response indicating that the access has failed, the returned read data must be zeroed and its write data ignored. When **RAZ/WI** occurs, a OK response is returned indicating that the access is successfully completed except that the read data is always zeroed and write data is ignored.

Some regions of memory map are reserved to maintain compatibility with past and future subsystems. Other areas are mapped to Expansion interfaces.

All accesses targeting populated volatile memory regions within `0x2100_0000` to `0x21FF_FFFF`, and `0x3100_0000` to `0x31FF_FFFF` support exclusive access since they implement exclusive access monitoring, provided the accesses are from:

- The processors
- Expansion managers through the Subordinate main expansion interfaces and Subordinate peripheral expansion interfaces.

Exclusive access is not supported for other regions implemented within the subsystem. For regions that reside in user expansion areas, exclusive access support is defined by the user expansion logic. If an exclusive access tries to access a region that does not support exclusive accesses, these accesses are not monitored for exclusive access and might still update their target memory locations regardless of their associated exclusive responses.

8.1.1 High level system address map

Security values do not define privileged or unprivileged accessibility. These are defined by the PPC or by the register blocks that is mapped to each area. See the individual areas for more details.

The system address map defines the following values for accessibility:

S

Secure Access

NS

Non-secure

NSC

Non-secure Callable

Note

The NSC values are defined through registers in the Secure Access Configuration Register block.

Table 8-1: High Level system address map

Row ID	Address	Size	Region	Alias	IDAU Region Values	Description
1	`0x00000000 - 0x00FFFFFF`	16MB	ITCM	5	• Security: NS • IDAUID: 0 • NSC: 0	CPU Instruction TCM. See CPU TCM memories.
2	`0x01000000 - 0x09FFFFFF`	144MB	Code Expansion	6	• Security: NS • IDAUID: 0 • NSC: 0	Manager Code Main Expansion Interface. See Main Interconnect Expansion Interfaces.
2.1	`0x0A000000 - 0x0AFFFFFF`	16MB	CPU0 ITCM	6.1	• Security: NS • IDAUID: 0 • NSC: 0	See TCM subordinate interface.
2.2	`0x0B000000 - 0x0BFFFFFF`	16MB	CPU1 ITCM	6.2	• Security: NS • IDAUID: 0 • NSC: 0	See TCM subordinate interface.
2.3	`0x0C000000 - 0x0CFFFFFF`	16MB	CPU2 ITCM	6.3	• Security: NS • IDAUID: 0 • NSC: 0	See TCM subordinate interface.

Row ID	Address	Size	Region	Alias	IDAU Region Values	Description
2.4	`0x0D000000` - `0x0DFFFFFF`	16MB	CPU3 ITCM	6.4	• Security: NS • IDAUID: 0 • NSC: 0	See TCM subordinate interface.
3	`0x0E000000` - `0x0E001FFF`	8KB	Reserved	-	• Security: NS • IDAUID: 0 • NSC: 0	Reserved
4	`0x0E002000` - `0x0FFFFFFF`	-	Reserved	-	• Security: NS • IDAUID: 0 • NSC: 0	Reserved
5	`0x10000000` - `0x10FFFFFF`	16MB	ITCM	1	• Security: S • IDAUID: 1 • NSC: CODENSC	CPU Instruction TCM. See CPU TCM memories.
5.1	`0x11000000` - `0x11FFFFFF`	16MB	ROM	-	• Security: S • IDAUID: 1 • NSC: CODENSC	Boot ROM. See ROM.
6	`0x12000000` - `0x19FFFFFF`	128MB	Code Expansion	2	• Security: S • IDAUID: 1 • NSC: CODENSC	Manager Code Main Expansion Interface. See Main Interconnect Expansion Interfaces.
6.1	`0x1A000000` - `0x1AFFFFFF`	16MB	CPU0 ITCM	2.1	• Security: S • IDAUID: 1 • NSC: CODENSC	See TCM subordinate interface.
6.2	`0x1B000000` - `0x1BFFFFFF`	16MB	CPU1 ITCM	2.2	• Security: S • IDAUID: 1 • NSC: CODENSC	See TCM subordinate interface.
6.3	`0x1C000000` - `0x1CFFFFFF`	16MB	CPU2 ITCM	2.3	• Security: S • IDAUID: 1 • NSC: CODENSC	See TCM subordinate interface.
6.4	`0x1D000000` - `0x1DFFFFFF`	16MB	CPU3 ITCM	2.4	• Security: S • IDAUID: 1 • NSC: CODENSC	See TCM subordinate interface.

Row ID	Address	Size	Region	Alias	IDAU Region Values	Description
7	`0x1E000000` - `0x1E001FFF`	8KB	Reserved	3	• Security: S • IDAUID: 1 • NSC: CODENSC	Reserved
8	`0x1E002000` - `0x1FFFFFFF`	-	Reserved	-	• Security: S • IDAUID: 1 • NSC: CODENSC	Reserved
9	`0x20000000` - `0x20FFFFFF`	16MB	DTCM	13	• Security: NS • IDAUID: 2 • NSC: 0	CPU Data TCM. See CPU TCM memories.
10	`0x21000000` - `0x21FFFFFF`	16MB	Volatile Memory	14	• Security: NS • IDAUID: 2 • NSC: 0	Volatile memory See Volatile memory region.
11	`0x22000000` - `0x23FFFFFF`	32MB	Reserved	-	• Security: NS • IDAUID: 2 • NSC: 0	Reserved
11.1	`0x24000000` - `0x24FFFFFF`	16MB	CPU0 DTCM	15.1	• Security: NS • IDAUID: 2 • NSC: 0	See TCM subordinate interface.
11.2	`0x25000000` - `0x25FFFFFF`	16MB	CPU1 DTCM	15.2	• Security: NS • IDAUID: 2 • NSC: 0	See TCM subordinate interface.
11.3	`0x26000000` - `0x26FFFFFF`	16MB	CPU2 DTCM	15.3	• Security: NS • IDAUID: 2 • NSC: 0	See TCM subordinate interface.
11.4	`0x27000000` - `0x27FFFFFF`	16MB	CPU3 DTCM	15.4	• Security: NS • IDAUID: 2 • NSC: 0	See TCM subordinate interface.
12	`0x28000000` - `0x2FFFFFFF`	128MB	Main Expansion	-	• Security: NS • IDAUID: 2 • NSC: 0	Manager Main Expansion Interface. See Main Interconnect Expansion Interfaces.

Row ID	Address	Size	Region	Alias	IDAU Region Values	Description
13	`0x30000000 - 0x30FFFFFF`	16MB	DTCM	9	• Security: S • IDAUID: 3 • NSC: RAMNSC	CPU Data TCM. See CPU TCM memories.
14	`0x31000000 - 0x31FFFFFF`	16MB	Volatile memory	10	• Security: S • IDAUID: 3 • NSC: RAMNSC	Internal multi-bank volatile memory. See Volatile memory region
15	`0x32000000 - 0x33FFFFFF`	32MB	Reserved	-	• Security: S • IDAUID: 3 • NSC: RAMNSC	Reserved
15.1	`0x34000000 - 0x34FFFFFF`	16MB	CPU0 DTCM	11.1	• Security: S • IDAUID: 3 • NSC: RAMNSC	See TCM subordinate interface.
15.2	`0x35000000 - 0x35FFFFFF`	16MB	CPU1 DTCM	11.2	• Security: S • IDAUID: 3 • NSC: RAMNSC	See TCM subordinate interface.
15.3	`0x36000000 - 0x36FFFFFF`	16MB	CPU2 DTCM	11.3	• Security: S • IDAUID: 3 • NSC: RAMNSC	See TCM subordinate interface.
15.4	`0x37000000 - 0x37FFFFFF`	16MB	CPU3 DTCM	11.4	• Security: S • IDAUID: 3 • NSC: RAMNSC	See TCM subordinate interface.
16	`0x38000000 - 0x3FFFFFFF`	128MB	Main Expansion	-	• Security: S • IDAUID: 3 • NSC: RAMNSC	Manager Main Expansion Interface. See Main Interconnect Expansion Interfaces.
17	`0x40000000 - 0x4000FFFF`	64KB	Peripherals	27	• Security: NS • IDAUID: 4 • NSC: 0	Peripheral Region. See Peripheral Region.
18	`0x40010000 - 0x4001FFFF`	64KB	Private CPU	-	• Security: NS • IDAUID: 4 • NSC: 0	CPU Private Peripheral Region. See CPU Private Region.

Row ID	Address	Size	Region	Alias	IDAU Region Values	Description
19	`0x40020000` - `0x4003FFFF`	128KB	System Control Peripheral Region	-	• Security: NS • IDAUID: 4 • NSC: 0	System Control Peripheral Region. See System Control Peripheral Region.
20	`0x40040000` - `0x400FFFFF`	768KB	Peripherals	-	• Security: NS • IDAUID: 4 • NSC: 0	Peripheral Region. See Peripheral Region.
21	`0x40100000` - `0x47FFFFFF`	127MB	Peripheral Expansion	-	• Security: NS • IDAUID: 4 • NSC: 0	Manager Peripheral Expansion Interface. See Main Interconnect Expansion Interfaces.
22	`0x48000000` - `0x4800FFFF`	64KB	Peripherals	32	• Security: NS • IDAUID: 4 • NSC: 0	Peripheral Region. See Peripheral Region.
23	`0x48010000` - `0x4801FFFF`	64KB	Private CPU	-	• Security: NS • IDAUID: 4 • NSC: 0	CPU Private Peripheral Region. See CPU Private Region.
24	`0x48020000` - `0x4803FFFF`	128KB	System Control	-	• Security: NS • IDAUID: 4 • NSC: 0	System Control Peripheral Region. See System Control Peripheral Region.
25	`0x48040000` - `0x480FFFFF`	768KB	Peripherals	-	• Security: NS • IDAUID: 4 • NSC: 0	Peripheral Region. See Peripheral Region.
26	`0x48100000` - `0x4FFFFFFF`	127MB	Peripheral Expansion	-	• Security: NS • IDAUID: 4 • NSC: 0	Manager Peripheral Expansion Interface. See Peripheral Interconnect Expansion Interfaces.
27	`0x50000000` - `0x5000FFFF`	64KB	Peripherals	17	• Security: S • IDAUID: 5 • NSC: 0	Peripheral Region. See Peripheral Region.
28	`0x50010000` - `0x5001FFFF`	64KB	Private CPU	-	• Security: S • IDAUID: 5 • NSC: 0	CPU Private Peripheral Region. See CPU Private Region.
29	`0x50020000` - `0x5003FFFF`	128KB	System Control	-	• Security: S • IDAUID: 5 • NSC: 0	System Control Peripheral Region. See System Control Peripheral Region.

Row ID	Address	Size	Region	Alias	IDAU Region Values	Description
30	`0x50040000 - 0x500FFFFF`	786KB	Peripherals	-	• Security: S • IDAUID: 5 • NSC: 0	Peripheral Region. See Peripheral Region.
31	`0x50100000 - 0x57FFFFFF`	127MB	Peripheral Expansion	-	• Security: S • IDAUID: 5 • NSC: 0	Manager Peripheral Expansion Interface. See Peripheral Interconnect Expansion Interfaces
32	`0x58000000 - 0x5800FFFF`	64KB	Peripherals	22	• Security: S • IDAUID: 5 • NSC: 0	Peripheral Region. See Peripheral Region.
33	`0x58010000 - 0x5801FFFF`	64KB	Private CPU	-	• Security: S • IDAUID: 5 • NSC: 0	Processor Private Peripheral Region. See CPU Private Region.
34	`0x58020000 - 0x5803FFFF`	128KB	System Control	-	• Security: S • IDAUID: 5 • NSC: 0	System Control Peripheral Region. See System Control Peripheral Region.
35	`0x58040000 - 0x580FFFFF`	768KB	Peripherals	-	• Security: S • IDAUID: 5 • NSC: 0	Peripheral Region. See Peripheral Region.
36	`0x58100000 - 0x5FFFFFFF`	127MB	Peripheral Expansion	-	• Security: S • IDAUID: 5 • NSC: 0	Manager Peripheral Expansion Interface. See Peripheral Interconnect Expansion Interfaces.
37	`0x60000000 - 0x6FFFFFFF`	256MB	Main Expansion	-	• Security: NS • IDAUID: 6 • NSC: 0	Manager Main Expansion Interface. See Main Interconnect Expansion Interfaces.
38	`0x70000000 - 0x7FFFFFFF`	256MB	Main Expansion	-	• Security: S • IDAUID: 7 • NSC: 0	Manager Main Expansion Interface. See Main Interconnect Expansion Interfaces.
39	`0x80000000 - 0x8FFFFFFF`	256MB	Main Expansion	-	• Security: NS • IDAUID: 8 • NSC: 0	Manager Main Expansion Interface. See Main Interconnect Expansion Interfaces.
40	`0x90000000 - 0x9FFFFFFF`	256MB	Main Expansion	-	• Security: S • IDAUID: 9 • NSC: 0	Manager Main Expansion Interface. See Main Interconnect Expansion Interfaces.
41	`0xA0000000 - 0xAFFFFFFF`	256MB	Main Expansion	-	• Security: NS • IDAUID: A • NSC: 0	Manager Main Expansion Interface. See Main Interconnect Expansion Interfaces.

Row ID	Address	Size	Region	Alias	IDAU Region Values	Description
42	`0xB0000000 - 0xBFFFFFFF`	256MB	Main Expansion	-	• Security: S • IDAUID: B • NSC: 0	Manager Main Expansion Interface. See Main Interconnect Expansion Interfaces.
43	`0xC0000000 - 0xCFFFFFFF`	256MB	Main Expansion	-	• Security: NS • IDAUID: C • NSC: 0	Manager Main Expansion Interface. See Main Interconnect Expansion Interfaces.
44	`0xD0000000 - 0xDFFFFFFF`	256MB	Main Expansion	-	• Security: S • IDAUID: D • NSC: 0	Manager Main Expansion Interface. See Main Interconnect Expansion Interfaces.
45	`0xE0000000 - 0xE00FFFFF`	1MB	PPB	-	Exempt	CPU Private Peripheral Bus Region. Local to the CPU. See CPU Private Peripheral Bus Region.
46	`0xE0100000 - 0xE01FFFFF`	1MB	Debug System	49	• Security: NS • IDAUID: E • NSC: 0	Debug System Access Region. See Debug System Access Region.
47	`0xE0200000 - 0xEFFFFFFF`	254MB	Peripheral Expansion	-	• Security: NS • IDAUID: E • NSC: 0	Manager Peripheral Expansion Interface. See Peripheral Interconnect Expansion Interfaces.
48	`0xF0000000 - 0xF00FFFFF`	1MB	Reserved	-	• Security: S • IDAUID: F • NSC: 0	Reserved
49	`0xF0100000 - 0xF01FFFFF`	1MB	Debug System	46	• Security: S • IDAUID: F • NSC: 0	Debug System Access Region. See Debug System Access Region.
50	`0xF0200000 - 0xFFFFFFFF`	254MB	Peripheral Expansion	-	• Security: S • IDAUID: F • NSC: 0	Manager Peripheral Expansion Interface. See Peripheral Interconnect Expansion Interfaces.

8.2 CPU TCM memories

The processors in the CRSAS Ma2 are configured to implement Tightly Coupled Memories (TCM) for Instruction and Data. These memories reside in the following location from the perspective of each CPU core:

- `0x0000_0000` to `0x00FF_FFFF` and `0x1000_0000` to `0x10FF_FFFF` for Instruction TCM. Both regions are aliased, and each provides up to 16MB of TCM space.

- `0x2000_0000` to `0x20FF_FFFF` and `0x3000_0000` to `0x30FF_FFFF` for Data TCM. Both regions are aliased, and each provides up to 16MB of TCM space.

Each CPU has access to its own local TCMs through its private address and other CPUs TCMs through the Main Interconnect. A CPU does not have access to its own TCMs through the Main Interconnect. Other managers on the Main interconnect have access to the TCMs. A TCM DMA subordinate interface to allow expansion managers to access the TCMs is provided. For more information, see TCM subordinate interface.

Note

While a CPU can directly access the TCM memory of another CPU through the Main Interconnect using the remapped TCM regions, These TCMs are in reality provided only for use by the CPU that owns the TCM. As a result, these TCMs should not be used as shared memory between CPUs. If there is a need to move data from one TCMs to another, we recommend either using DMA, or software emulating DMA, to move blocks of data from one TCM to another for processing after determining that it is safe to move the data.

All TCMs always start at the base address of each region. Any memory areas in that region that are not used are reserved and return a bus error response if accessed.

8.3 ROM

The CRSAS Ma2 supports an optional ROM bank that contains the First Stage Boot Loader (FSBL).

The ROM exist when ROMADDRWIDTH is not zero and the ROM size is defined according to ROMADDRWIDTH in the Configuration options. The ROM is not aliased, that is, used for Secure access only, and located within the ROM region as defined in System memory map overview.

The ROM resides in the following location within a 16MB ROM region:

- `0x1100_0000` to `0x11FF_FFFF` for Secure access

Any memory areas in that region that are not utilised are reserved and return a bus error response if accessed.

If the ROM does not exist, the region `0x1100_0000` to `0x11FF_FFFF` is mapped to the Manager Code Main Expansion interface. See Main Interconnect Expansion interfaces.

The ROM has a Main Interconnect Peripheral Protection Controller (MIPPC0) associated with it that provides protection on subordinate side and restricts the ROM to Secure privileged world. The PPC protected components can raise interrupt on security violation and respond with bus error or **RAZ/WI** depending on the global SECRESPCFG configuration.

8.4 Volatile memory region

The CRSAS Ma2 supports up to four internal Volatile Memory (VM) Banks. While these are typically implemented as SRAMs, the actual memories used are **IMPLEMENTATION DEFINED**.

All VM banks in the system are of the same size and collectively they form a contiguous area of memory of up to 16MB. These are then aliased onto both the Secure and Non-secure memory regions. A Memory protection controller per VM then divides the VM into pages and determines where each page resides in either the Secure or Non-secure regions. Any unused areas within that 16MB region are reserved.

The [@#tbl:volmemreg_addressmap] shows an example where two memory banks are configured. In the example each memory bank is 512kB in size, adding up to 1MB. The remaining 15MB is reserved.

The Volatile memory region example address map defines the following values for security:

NS_MPC

Non-secure access only gated by an MPC.

S_MPC

Secure access only gated by an MPC.

Table 8-2: Volatile memory region example address map

Row ID	From address	To address	Size	Region name	Alias with row ID	Security	Description
1	`0x2100_0000`	`0x2107_FFFF`	512KB	VM0	4	NS_MPC	Maps to Internal Volatile Memory Bank 1
2	`0x2108_0000`	`0x210F_FFFF`	512KB	VM1	5	NS_MPC	Maps to Internal Volatile Memory Bank 2
3	`0x2110_2000`	`0x21FF_FFFF`	15MB	Reserved	-	-	Reserved
4	`0x3100_0000`	`0x3107_FFFF`	512KB	VM0	1	S_MPC	Maps to Internal Volatile Memory Bank 1
5	`0x3108_0000`	`0x310F_FFFF`	512KB	VM1	2	S_MPC	Maps to Internal Volatile Memory Bank 2
8	`0x3110_2000`	`0x31FF_FFFF`	15MB	Reserved	-	-	Reserved

8.5 Peripheral Region

The Peripheral Region are memory regions where the peripherals of the system reside. There are eight regions in total:

`0x4000_0000` to `0x4000_FFFF`

Non-secure region for low latency peripherals that are expected to be aliased in its associated Secure region, `0x5000_0000` to `0x5000_FFFF`.

`0x4004_0000` to `0x400F_FFFF`

Non-secure region for low latency peripherals that are expected to be not aliased.

`0x4800_0000` to `0x4800_FFFF`

Non-secure region for high latency peripherals that are expected to be aliased in its associated Secure region, `0x5800_0000` to `0x5800_FFFF`.

`0x4804_0000` to `0x480F_FFFF`
: Non-secure region for high latency peripherals that are expected to be not aliased.

`0x5000_0000` to `0x5000_FFFF`
: Secure region for low latency peripherals that are expected to be aliased in its associated Non-secure region, `0x4000_0000` to `0x4000_FFFF`.

`0x5004_0000` to `0x500F_FFFF`
: Secure region for low latency peripherals that are expected to be not aliased.

`0x5800_0000` to `0x5800_FFFF`
: Secure region for high latency peripherals that are expected to be aliased in its associated Non-secure region, `0x4800_0000` to `0x4800_FFFF`.

`0x5804_0000` to `0x580F_FFFF`
: Secure region for high latency peripherals that are expected to be not aliased.

For regions that are aliased to both a Secure and a Non-secure region, the final mapping of a peripheral in these regions to either a Secure or a Non-secure region is either determined by the PPC that is programmed using Secure Access Configuration registers, or determined by the peripheral that intrinsically supports TrustZone.

The following table shows the memory map of the Peripheral Regions.

The Peripheral Region address map defines the following values for security:

NS_PPC
: Non-secure access only, gated by a PPC.

S_PPC
: Secure access only, gated by a PPC.

S
: Secure access only.

NS
: Non-secure access only.

P
: Privileged access only

UP
: Unprivileged and privileged access allowed

P_PPC
: Unprivileged access controlled by PPC

When PPC protects a peripheral, the configurability of the security or privileged attributes for a given peripheral is defined by PERIPHSPPPC0, PERIPHSPPPC1, PERIPHNSPPPC0, PERIPHNSPPPC1, NPUSPPORSL, NPUNSPORPL, NPUSPPORPL, PERIPHNSPPC0, and PERIPHNSPPC1 registers.

Table 8-3: Peripheral Region address map

Row ID	From address	To address	Size	Region name	Alias with row ID	Security	Description
1	`0x4000_0000`	`0x4000_0FFF`	4KB	MHU0	23	NS_PPC, P_PPC	Message Handling Unit 0. Only exists if NUMCPU > 0. See Message Handling Unit.
2	`0x4000_1000`	`0x4000_1FFF`	4KB	MHU1	24	NS_PPC, P_PPC	Message Handling Unit 1. Only exists if NUMCPU > 0. See Message Handling Unit.
3	`0x4000_2000`	`0x4000_3FFF`	8KB	DMA	25	NS, UP	DMA Configuration interface Only exists if NUMDMA = 1 See DMA registers.
4	`0x4000_4000`	`0x4000_4FFF`	4KB	NPU0	26	NS_PPC, P_PPC	NPU0 Configuration interface Only exists if NUMNPU > 0. See NPU<m> registers.
5	`0x4000_5000`	`0x4000_5FFF`	4KB	NPU1	27	NS_PPC, P_PPC	NPU1 Configuration interface Only exists if NUMNPU > 0. See NPU<m> registers.
6	`0x4000_6000`	`0x4000_6FFF`	4KB	NPU2	28	NS_PPC, P_PPC	NPU2 Configuration interface Only exists if NUMNPU > 0. See NPU<m> registers.
7	`0x4000_7000`	`0x4000_7FFF`	4KB	NPU3	29	NS_PPC, P_PPC	NPU3 Configuration interface Only exists if NUMNPU > 0. See NPU<m> registers.
8	`0x4000_8000`	`0x4000_FFFF`	-	Reserved	-	-	Reserved (RAZ/WI)
9	`0x4004_0000`	`0x4007_FFFF`	-	Reserved	-	-	Reserved
10	`0x4008_0000`	`0x4008_0FFF`	4KB	NSACFG	-	NS_PPC, P, P_PPC	Non-secure Access Configuration Register block See Non-secure access configuration register block.
11	`0x4008_1000`	`0x4008_FFFF`	-	Reserved	-	-	Reserved (RAZ/WI)
12	`0x4009_0000`	`0x4009_3FFF`	16KB	Reserved	-	-	Reserved
13	`0x4009_4000`	`0x400F_FFFF`	-	Reserved	-	-	Reserved
14	`0x4800_0000`	`0x4800_0FFF`	4KB	TIMER0	43	NS_PPC P_PPC	Timer 0. See Timestamp-based timers registers.

Row ID	From address	To address	Size	Region name	Alias with row ID	Security	Description
15	`0x4800_1000`	`0x4800_1FFF`	4KB	TIMER1	44	NS_PPC P_PPC	Timer 1. See Timestamp-based timers registers.
16	`0x4800_2000`	`0x4800_2FFF`	4KB	TIMER2	45	NS_PPC P_PPC	Timer 2. See Timestamp-based timers registers.
17	`0x4800_3000`	`0x4800_3FFF`	4KB	TIMER3	46	NS_PPC P_PPC	Timer 3. See Timestamp-based timers registers.
	`0x4800_4000`	`0x4800_EFFF`	-	Reserved	-	-	Reserved (RAZ/WI)
18	`0x4800_F000`	`0x4800_FFFF`	-	SDC-600	47	NS_PPC P_PPC	SDC-600 Internal APBCOM. See Secure Debug Channel registers. If SDC-600 does not exist, this area is Reserved (RAZ/WI)
19	`0x4804_0000`	`0x4804_0FFF`	4KB	NSWDCTRL	-	NS_PPC, P_PPC, P	Non-secure Watchdog Control Frame. See Timestamp-based Watchdogs registers.
20	`0x4804_1000`	`0x4804_1FFF`	4KB	NSWDREF	-	NS_PPC, P_PPC	Non-secure Watchdog Refresh Frame. See Timestamp-based Watchdogs registers.
21	`0x4804_2000`	`0x4804_FFFF`	-	Reserved	-	-	Reserved (RAZ/WI)
22	`0x4805_0000`	`0x480F_FFFF`	-	Reserved	-	-	Reserved
23	`0x5000_0000`	`0x5000_0FFF`	4KB	MHU0	1	S_PPC, P_PPC	Message Handling Unit 0. Only exists if NUMCPU > 0. See Message Handling Unit.
24	`0x5000_1000`	`0x5000_1FFF`	4KB	MHU1	2	S_PPC, P_PPC	Message Handling Unit 1. Only exists if NUMCPU > 0. See Message Handling Unit.
25	`0x5000_2000`	`0x5000_3FFF`	8KB	DMA	3	S, UP	DMA Configuration interface Only exists if NUMDMA = 1 See DMA registers.
26	`0x5000_4000`	`0x5000_4FFF`	4KB	NPU0	4	S_PPC, P_PPC	NPU0 Configuration interface Only exists if NUMNPU > 0. See NPU<m> registers.
27	`0x5000_5000`	`0x5000_5FFF`	4KB	NPU1	5	S_PPC, P_PPC	NPU1 Configuration interface Only exists if NUMNPU > 0. See NPU<m> registers.

Row ID	From address	To address	Size	Region name	Alias with row ID	Security	Description
28	`0x5000_6000`	`0x5000_6FFF`	4KB	NPU2	6	S_PPC, P_PPC	NPU2 Configuration interface Only exists if NUMNPU > 0. See NPU<m> registers.
29	`0x5000_7000`	`0x5000_7FFF`	4KB	NPU3	7	S_PPC, P_PPC	NPU3 Configuration interface Only exists if NUMNPU > 0. See NPU<m> registers.
30	`0x5000_8000`	`0x5000_FFFF`	-	Reserved	-	-	Reserved (RAZ/WI)
31	`0x5004_0000`	`0x5007_FFFF`	-	Reserved	-	-	Reserved
32	`0x5008_0000`	`0x5008_0FFF`	4KB	SACFG	-	S_PPC, P_PPC, P	Secure Access Configuration Register block Secure access configuration register block.
33	`0x5008_1000`	`0x5008_2FFF`	-	Reserved	-	-	Reserved (RAZ/WI)
34	`0x5008_3000`	`0x5008_3FFF`	4KB	VM0MPC	-	S_PPC, P_PPC, P	VM0 Memory Protection Controller. The number of VM<i>MPC regions depends on NUMVMBANK. If VM<i> does not exist, then the VM<i>MPC region is reserved. See Volatile memory.
35	`0x5008_4000`	`0x5008_4FFF`	4KB	VM1MPC	-	S_PPC, P_PPC, P	VM1 Memory Protection Controller. The number of VM<i>MPC regions depends on NUMVMBANK. If VM<i> does not exist, then the VM<i>MPC region is reserved. See Volatile memory.
36	`0x5008_5000`	`0x5008_5FFF`	4KB	VM2MPC	-	S_PPC, P_PPC, P	VM2 Memory Protection Controller. The number of VM<i>MPC regions depends on NUMVMBANK. If VM<i> does not exist, then the VM<i>MPC region is reserved. See Volatile memory.
37	`0x5008_6000`	`0x5008_6FFF`	4KB	VM3MPC	-	S_PPC, P_PPC, P	VM3 Memory Protection Controller. The number of VM<i>MPC regions depends on NUMVMBANK. If VM<i> does not exist, then the VM<i>MPC region is reserved. See Volatile memory.
38	`0x5008_7000`	`0x5009_DFFF`	-	Reserved	-	-	Reserved (RAZ/WI)
39	`0x5009_E000`	`0x5009_EFFF`	4KB	KMU	-	S_PPC, P_PPC,	Key Management Unit When LCM_KMU_SAM_PRESENT = 0 this is Reserved.
40	`0x5009_F000`	`0x5009_FFFF`	4KB	Reserved	-	-	Reserved

Row ID	From address	To address	Size	Region name	Alias with row ID	Security	Description
41	`0x500A_0000`	`0x500A_FFFF`	64KB	LCM	-	S_PPC, P_PPC,	Lifecycle Manager When LCM_KMU_SAM_PRESENT = 0 this is Reserved.
42	`0x500B_0000`	`0x500F_FFFF`	-	Reserved	-	-	Reserved
43	`0x5800_0000`	`0x5800_0FFF`	4KB	TIMER0	14	S_PPC, P_PPC	Timer 0. See Timestamp-based timers registers.
44	`0x5800_1000`	`0x5800_1FFF`	4KB	TIMER1	15	S_PPC, P_PPC	Timer 1. See Timestamp-based timers registers.
45	`0x5800_2000`	`0x5800_2FFF`	4KB	TIMER2	16	S_PPC, P_PPC	Timer 2. See Timestamp-based timers registers.
46	`0x5800_3000`	`0x5800_3FFF`	4KB	TIMER3	17	S_PPC, P_PPC	Timer 3. See Timestamp-based timers registers.
	`0x5800_4000`	`0x5800_EFFF`	-	Reserved	-	-	Reserved (RAZ/WI)
47	`0x5800_F000`	`0x5800_FFFF`	-	SDC-600	18	S_PPC P_PPC	SDC-600 Internal APBCOM. See Secure Debug Channel registers. If SDC-600 does not exist, this area is Reserved and RAZWI
48	`0x5804_0000`	`0x5804_0FFF`	4KB	SWDCTRL	-	S_PPC, P_PPC, P	Secure Watchdog Control Frame. See Timestamp-based Watchdogs registers.
49	`0x5804_1000`	`0x5804_1FFF`	4KB	SWDREF	-	S_PPC, P_PPC	Secure Watchdog Refresh Frame. See Timestamp-based Watchdogs registers.
50	`0x5804_2000`	`0x5804_2FFF`	4KB	SAM	-	S_PPC, P_PPC,	Security Alarm Manager When LCM_KMU_SAM_PRESENT = 0 this is Reserved.
51	`0x5804_3000`	`0x5804_FFFF`	-	Reserved	-	-	Reserved (RAZ/WI)
52	`0x5805_0000`	`0x580F_FFFF`	-	Reserved	-	-	Reserved

8.5.1 Message Handling Unit register map

The CRSAS Ma2 implements up to two MHUs. These allow thesoftware to raise interrupts to the CPU cores. Both MHUs are mapped twice into both Secure and Non-secure regions as follows, and a PPC then controls which area each MHU resides:

- MHU0 in Non-secure region at `0x4000_0000` and Secure region at `0x5000_0000`
- MHU1 in Non-secure region at `0x4000_1000` and Secure region at `0x5000_1000`

If there is only one CPU core in the system, neither MHU exists and the two regions are reserved and any accesses to an MHU results in **RAZ/WI**. For write access to these registers, only 32-bit writes are supported. Any byte and halfword writes result in its write data ignored.

Note

There are two MHUs in the system so that the software can map one for Secure use and another for Non-secure use. However, the software can choose to map both to Secure, or Non-secure if needed.

Each MHU has the Table 8-4: Message Handling Unit register map on page 176. The security of each MHU register area is defined by the drive of the PPCs by registers in Secure Access Configuration Register block and Non-secure Access Configuration Register block. See PERIPHNSPPC0, PERIPHNSPPC1, PERIPHSPPPC0, PERIPHSPPPC1, PERIPHNSPPPC0, and PERIPHNSPPPC1.

All registers reside in the PD_SYS power domain and are reset by nWARMRESETSYS. The CRSAS Ma2 defines up to two MHUs.

Note

In Table 8-4: Message Handling Unit register map on page 176, the following set of registers per processor exists:

- CPU<*n*>INTR_STAT
- CPU<*n*>INTR_SET
- CPU<*n*>INTR_CLR

Table 8-4: Message Handling Unit register map

Offset	Name	Type	Reset	Description
`0x000` + *n*(`0x010`)	CPU<*n*>INTR_STAT	RO	`0x0000_0000`	CPU <*n*> Interrupt Status Register. See CPU<n>INTR_STAT.
`0x004` + *n*(`0x010`)	CPU<*n*>INTR_SET	WO	`0x0000_0000`	CPU <*n*> Interrupt Set Register. See CPU<n>INTR_SET.
`0x008` + *n*(`0x010`)	CPU<*n*>INTR_CLR	WO	`0x0000_0000`	CPU <*n*> Interrupt Clear Register. See CPU<n>INTR_CLR.
`0x00C` + *n*(`0x010`)		RAZ/WI	`0x0000_0000`	Reserved
(NUMCPU+1)(`0x010`) – `0xFC8`		RAZ/WI	`0x0000_0000`	Reserved
`0xFD0`	PIDR4	RO	`0x0000_0004`	Peripheral ID 4
`0xFD4` – `0xFDC`		RAZ/WI	`0x0000_0000`	Reserved
`0xFE0`	PIDR0	RO	`0x0000_0056`	Peripheral ID 0
`0xFE4`	PIDR1	RO	`0x0000_00B8`	Peripheral ID 1
`0xFE8`	PIDR2	RO	`0x0000_001B`	Peripheral ID 2
`0xFEC`	PIDR3	RO	`0x0000_0000`	Peripheral ID 3
`0xFF0`	CIDR0	RO	`0x0000_000D`	Component ID 0
`0xFF4`	CIDR1	RO	`0x0000_00F0`	Component ID 1
`0xFF8`	CIDR2	RO	`0x0000_0005`	Component ID 2
`0xFFC`	CIDR3	RO	`0x0000_00B1`	Component ID 3

8.5.1.1 CPU <n> Interrupt registers

The CPU <n> Interrupt registers, CPU<*n*>INTR_STAT, CPU<*n*>INTR_SET, and CPU<*n*>INTR_CLR, allow software to raise an interrupt, clear an interrupt and check the value written that is used to raise the interrupt to CPU <*n*>. Separate Set and Clear registers allow individual bit of the interrupt status to be set and cleared, therefore it supports SW using each bit to represent an event that can be independently set and cleared. One set of these registers exist per CPU<*n*>.

8.5.1.1.1 CPU<n>INTR_STAT

CPU <n> Interrupt Status register.

Configurations

Present only when NUMCPU > 0.

This register is **RAZ/WI** if NUMCPU =0.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

For write access to this register, only 32-bit writes are supported. Any byte and halfword writes result in its write data ignored.

Bit descriptions

Table 8-5: CPU<n>INTR_STAT bit descriptions

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	ROWI	`0x000_0000`
[3:0]	CPU<n> INTR_STAT	CPU <*n*> Interrupt Status. When any bit is set to HIGH, the MHU interrupt signal to CPU <*n*> is set to HIGH.	ROWI	`0x0`

8.5.1.1.2 CPU<n>INTR_SET

The CPU <n> Interrupt Set register.

Configurations

Present only when NUMCPU > 0.

This register is **RAZ/WI** if NUMCPU =0.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

For write access to this register, only 32-bit writes are supported. Any byte and halfword writes result in its write data ignored.

Bit descriptions

Table 8-6: CPU<n>INTR_SET bit descriptions

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RAZ/WI	`0x000_0000`
[3:0]	CPU<n> INTR_SET	CPU <*n*> Interrupt Set. When a value '1' is written to CPU<*n*>INTR_SET[*i*], the corresponding CPU<*n*>INTR_STAT[*i*] is set to high.	RAZW1S	`0x0`

8.5.1.1.3 CPU<n>INTR_CLR

The CPU <n> Interrupt Clear register.

Configurations

Present only when NUMCPU > 0. This register is **RAZ/WI** if NUMCPU = 0.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

For write access to this register, only 32-bit writes are supported. Any byte and halfword writes result in its write data ignored.

Bit descriptions

Table 8-7: CPU<n>INTR_CLR bit descriptions

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RAZ/WI	`0x000_0000`
[3:0]	CPU<n> INTR_CLR	CPU *<n>* a value '1' is written to CPU<*n*>INTR_CLR[*i*], the corresponding CPU<*n*>INTR_STAT[*i*] is cleared.	RAZ/ W1C	`0x0`

8.5.2 Secure Access Configuration Register block

The Secure Access Configuration Register block implements program visible states that allow software to control security gating units within the design. The register block base address is `0x5008_0000`. These registers are Secure privileged access only and support 32-bit RW accesses. For write access to these registers, only 32-bit writes are supported. Any byte and halfword writes are ignored.

All registers reside in the PD_SYS power domain and are reset by nWARMRESETSYS.

The following table list the registers within this block. Details of each register are described in the following subsections.

Table 8-8: Secure Access Configuration Register block Register Map

Offset	Name	Type	Reset	Description
`0x000`	SPCSECCTRL	RW	`0x0000_0000`	Secure Privileged Controller Secure Configuration Control register, see SPCSECCTRL.
`0x004`	BUSWAIT	RW	CFG_DEF	Bus Access wait control after reset, BUSWAIT.
`0x008`	-	RAZ/ WI	`0x0000_0000`	Reserved
`0x010`	SECRESPCFG	RW	`0x0000_0000`	Security Violation Response Configuration Register, see SECRESPCFG.
`0x014`	NSCCFG	RW	`0x0000_0000`	Non-secure Callable Configuration for IDAU, see NSCCFG.
`0x018`	-	RAZ/ WI	`0x0000_0000`	Reserved
`0x01C`	SECMPCINTSTAT	RO	`0x0000_0000`	Secure MPC Interrupt Status, see SECMPCINTSTAT.
`0x020`	SECPPCINTSTAT	RO	`0x0000_0000`	Secure PPC Interrupt Status, see SECPPCINTSTAT.
`0x024`	SECPPCINTCLR	RW	`0x0000_0000`	Secure PPC Interrupt Clear, see SECPPCINTCLR.
`0x028`	SECPPCINTEN	RW	`0x0000_0000`	Secure PPC Interrupt Enable, see SECPPCINTEN.
`0x02C`	-	RAZ/ WI	`0x0000_0000`	Reserved
`0x030`	SECMSCINTSTAT	RO	`0x0000_0000`	Secure MSC Interrupt Status, see SECMSCINTSTAT.
`0x034`	SECMSCINTCLR	RW	`0x0000_0000`	Secure MSC Interrupt Clear, see SECMSCINTCLR.
`0x038`	SECMSCINTEN	RW	`0x0000_0000`	Secure MSC Interrupt Enable, see SECMSCINTEN.
`0x03C`	-	RAZ/ WI	`0x0000_0000`	Reserved
`0x040`	BRGINTSTAT	RO	`0x0000_0000`	Bridge Buffer Error Interrupt Status, see BRGINTSTAT.
`0x044`	BRGINTCLR	RW	`0x0000_0000`	Bridge Buffer Error Interrupt Clear, see BRGINTCLR.

Offset	Name	Type	Reset	Description
0x048	BRGINTEN	RW	0x0000_0000	Bridge Buffer Error Interrupt Enable, see BRGINTEN.
0x04C	-	RAZ/WI	0x0000_0000	Reserved
0x050	MAINNSPPC0	RW	0x0000_0000	Non-secure Access Peripheral Protection Control 0 on the Main Interconnect, see MAINNSPPC0.
0x054 -0x05C	-	RAZ/WI	0x0000_0000	Reserved
0x060	MAINNSPPCEXP0	RW	0x0000_0000	Expansion 0 Non-secure Access Peripheral Protection Control on the Main Interconnect, see MAINNSPPCEXP{0-3}.
0x064	MAINNSPPCEXP1	RW	0x0000_0000	Expansion 1 Non-secure Access Peripheral Protection Control on the Main Interconnect, see MAINNSPPCEXP{0-3}.
0x068	MAINNSPPCEXP2	RW	0x0000_0000	Expansion 2 Non-secure Access Peripheral Protection Control on the Main Interconnect, see MAINNSPPCEXP{0-3}.
0x06C	MAINNSPPCEXP3	RW	0x0000_0000	Expansion 3 Non-secure Access Peripheral Protection Control on the Main Interconnect, see MAINNSPPCEXP{0-3}.
0x070	PERIPHNSPPC0	RW	0x0000_0000	Non-secure Access Peripheral Protection Control 0 on Peripheral Interconnect. See PERIPHNSPPC0.
0x074	PERIPHNSPPC1	RW	0x0000_0000	Non-secure Access Peripheral Protection Control 1 on Peripheral Interconnect, see PERIPHNSPPC1.
0x078	NPUSPPORSL	RW	NPU<m>PORSLRST	NPU power on reset security level control. Each Field Controls the PORSL inputs for an associated NPU, see NPUSPPORSL.
0x07C	-	RAZ/WI	0x0000_0000	Reserved
0x080	PERIPHNSPPCEXP0	RW	0x0000_0000	Expansion 0 Non-secure Access Peripheral Protection Control on Peripheral Bus, see PERIPHNSPPCEXP{0-3}.
0x084	PERIPHNSPPCEXP1	RW	0x0000_0000	Expansion 1 Non-secure Access Peripheral Protection Control on Peripheral Bus, see PERIPHNSPPCEXP{0-3}.
0x088	PERIPHNSPPCEXP2	RW	0x0000_0000	Expansion 2 Non-secure Access Peripheral Protection Control on Peripheral Bus, see PERIPHNSPPCEXP{0-3}.
0x08C	PERIPHNSPPCEXP3	RW	0x0000_0000	Expansion 3 Non-secure Access Peripheral Protection Control on Peripheral Bus, see PERIPHNSPPCEXP{0-3}.
0x090	MAINSPPPC0	RW	0x0000_0000	Secure Unprivileged Access Peripheral Protection Control 0 on Main Interconnect, see MAINSPPPC0.
0x094 -0x09C	-	RAZ/WI	0x0000_0000	Reserved
0x0A0	MAINSPPPCEXP0	RW	0x0000_0000	Expansion 0 Secure Unprivileged Access Peripheral Protection Control on Main Interconnect, see MAINSPPPCEXP{0-3}.
0x0A4	MAINSPPPCEXP1	RW	0x0000_0000	Expansion 1 Secure Unprivileged Access Peripheral Protection Control on Main Interconnect, see MAINSPPPCEXP{0-3}.
0x0A8	MAINSPPPCEXP2	RW	0x0000_0000	Expansion 2 Secure Unprivileged Access Peripheral Protection Control on Main Interconnect, see MAINSPPPCEXP{0-3}.
0x0AC	MAINSPPPCEXP3	RW	0x0000_0000	Expansion 3 Secure Unprivileged Access Peripheral Protection Control on Main Interconnect, see MAINSPPPCEXP{0-3}.
0x0B0	PERIPHSPPPC0	RW	0x0000_0000	Secure Unprivileged Access Peripheral Protection Control 0 on Peripheral Interconnect, see PERIPHSPPPC0.

Offset	Name	Type	Reset	Description
0x0B4	PERIPHSPPPC1	RW	0x0000_0000	Secure Unprivileged Access Peripheral Protection Control 1 on Peripheral Interconnect, see PERIPHSPPPC1.
0x0B8	NPUSPPORPL	RW	NPU<m>PORPLRST	NPU power on reset Secure privilege level control. Each Field Controls the PORPL inputs of the associated NPU. see NPUSPPORPL.
0x0BC	-	RAZ/WI	0x0000_0000	Reserved
0x0C0	PERIPHSPPPCEXP0	RW	0x0000_0000	Expansion 0 Secure Unprivileged Access Peripheral Protection Control on Peripheral Interconnect, see PERIPHSPPPCEXP{0-3}.
0x0C4	PERIPHSPPPCEXP1	RW	0x0000_0000	Expansion 1 Secure Unprivileged Access Peripheral Protection Control on Peripheral Interconnect, see PERIPHSPPPCEXP{0-3}.
0x0C8	PERIPHSPPPCEXP2	RW	0x0000_0000	Expansion 2 Secure Unprivileged Access Peripheral Protection Control on Peripheral Interconnect, see PERIPHSPPPCEXP{0-3}.
0x0CC	PERIPHSPPPCEXP3	RW	0x0000_0000	Expansion 3 Secure Unprivileged Access Peripheral Protection Control on Peripheral Interconnect, see PERIPHSPPPCEXP{0-3}.
0x0D0	NSMSCEXP	RW	CFG_DEF	Expansion MSC Non-secure Configuration, see NSMSCEXP.
0x0D4 -0xFCC	-	RAZ/WI	0x0000_0000	Reserved
0xFD0	PIDR4	RO	0x0000_0004	Peripheral ID 4
0xFD4 -0xFDC	-	RAZ/WI	0x0000_0000	Reserved
0xFE0	PIDR0	RO	0x0000_0052	Peripheral ID 0
0xFE4	PIDR1	RO	0x0000_00B8	Peripheral ID 1
0xFE8	PIDR2	RO	0x0000_003B	Peripheral ID 2
0xFEC	PIDR3	RO	0x0000_0000	Peripheral ID 3
0xFF0	CIDR0	RO	0x0000_000D	Component ID 0
0xFF4	CIDR1	RO	0x0000_00F0	Component ID 1
0xFF8	CIDR2	RO	0x0000_0005	Component ID 2
0xFFC	CIDR3	RO	0x0000_00B1	Component ID 3

8.5.2.1 SPCSECCTRL

The Security Privilege Controller Security Configuration Control Register implements the security lock register.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-9: SPCSECCTRL bit descriptions

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved	RAZ/WI	`0x0000_0000`
[0]	SPCSECCFGLOCK	Active-HIGH Control to Disable writes to Security related control registers in the Secure Access Configuration Register block. When set to HIGH, it can no longer be cleared to zero except through reset or the PD_SYS turning OFF. Registers that can no longer be modified when SPCSECCFGLOCK is set to HIGH are: • NSCCFG • MAINNSPPC0 • MAINNSPPCEXP<i> • PERIPHNSPPC0 • PERIPHNSPPC1 • PERIPHNSPPCEXP<i> • MAINSPPPC0 • MAINSPPPCEXP<i> • PERIPHSPPPC0 • PERIPHSPPPC1 • PERIPHSPPPCEXP<i> • NSMSCEXP • NPUSPPORSL • NPUSPPORPL	RW1S	`0x0`

8.5.2.2 BUSWAIT

The Bus Access Wait Register allows software to gate access entering the interconnect from specific managers in the system, causing them to stall so that the CPU can complete the configuration of the MPCs or other Security registers in the system before the stalled accesses commencing.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-10: BUSWAIT bit descriptions

Bits	Name	Description	Type	Reset
[31:17]	-	Reserved	RAZ/WI	`0x0000`
[16]	ACC_WAITN_STATUS	This status register indicates the status of any gating units that are used to block bus access to the system: • '1': allow access. • '0': block access. This register reflects the combined status of all gating units in the system, including status on the input signal, ACCWAITNSTATUS expected to be driven from external gating units. Both ACC_WAITN_STATUS and ACC_WAITN together, ensuring that software can determine that all gates have reached the state that is requested.	RO	`0x0`
[15:1]	-	Reserved	RAZ/WI	`0x0000`
[0]	ACC_WAITN	Request gating units to block bus access to system: • '1': allow access. • '0': block access. This control only affects the Access Control Gates (ACG) in the system that feeds into the interconnect, and it excludes access from CPU cores. This register also drives the output signal ACCWAITn. Both ACC_WAITN_STATUS and ACC_WAITN together, ensuring that software can determine that all gates have reached the state that is requested.	RW	ACCWAITNRST

8.5.2.3 SECRESPCFG

The Security Violation Response Configuration Register is used to define the response to an access that causes security violation on the Bus Fabric.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-11: SECRESPCFG bit descriptions

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved	RAZ/WI	`0x0000_0000`
[0]	SECRESPCFG	This field configures the response in case of a security violation: • '0': Read-Zero Write Ignore • '1': Bus error Some components, for example, the AHB Memory Protection Controllers (MPC), provide their own control registers to configure their own response. Components that have their own register to control funtonality do not have to use this register.	RW	`0x0`

8.5.2.4 NSCCFG

The Non-secure Callable Configuration register allows software to define if the region `0x1000_0000` to `0x1FFF_FFFF` that normally hosts Secure code, and the region `0x3000_0000` to `0x3FFF_FFFF` that normally implements Secure Volatile Memories, are Non-secure Callable regions of memory.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-12: NSCCFG bit descriptions

Bits	Name	Description	Type	Reset
[31:2]	-	Reserved	RAZ/WI	`0x0000_0000`
[1]	RAMNSC	Configures if the region `0x3000_0000` to `0x3FFF_FFFF` is Non-secure Callable: • ‘0’: Not Non-secure Callable • ‘1’: Non-secure Callable.	RW	`0x0`
[0]	CODENSC	Configures if the CODE region `0x1000_0000` to `0x1FFF_FFFF` is Non-secure Callable: • ‘0’: Not Non-secure Callable • ‘1’: Non-secure Callable.	RW	`0x0`

8.5.2.5 SECMPCINTSTAT

The interrupt signals from all Memory Protection Controllers (MPC), both within the CRSAS Ma2 and in the Expansion logic, are merged and sent to the CPUs on a single Interrupt signal. The Secure MPC Interrupt Status Register therefore provides Secure software with the ability to check which one of the MPCs is causing the interrupt. When the source of the interrupt is identified, you must use the MPC register interface to clear the interrupt.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-13: SECMPCINTSTAT bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	SMPCEXP_STATUS	Interrupt Status for Expansion Memory Protection Controller. Each bit *i* shows the status of input signal SMPCEXPSTATUS[i] The MPCEXPDIS configuration point defines if each bit within this register is actually implemented such that if MPCEXPDIS[i] = 1'b1, then SMPCEXP_STATUS[i] is disabled and is **RAZ/WI**.	RO	`0x0000_0000`
[15:4]	-	Reserved	RAZ/WI	`0x0000_0000`
[3]	SMPCVM3_STATUS	Interrupt Status for Memory Protection Controller of Volatile Memory Bank 3. This register is not used and **RAZ/WI** if NUMVMBANK < 4	RO	`0x0`
[2]	SMPCVM2_STATUS	Interrupt Status for Memory Protection Controller of Volatile Memory Bank 2. This register is not used and **RAZ/WI** if NUMVMBANK < 3	RO	`0x0`
[1]	SMPCVM1_STATUS	Interrupt Status for Memory Protection Controller of Volatile Memory Bank 1. This register is not used and **RAZ/WI** if NUMVMBANK < 2	RO	`0x0`
[0]	SMPCVM0_STATUS	Interrupt Status for Memory Protection Controller of Volatile Memory Bank 0. This register is not used and **RAZ/WI** if NUMVMBANK < 1	RO	`0x0`

8.5.2.6 SECPPCINTSTAT

When access violations occur on any Peripheral Protection Controller, a level interrupt is raised through a combined interrupt that is then sent to the CPUs. The PPC Secure PPC Interrupt Status Register allows software to determine the source of the interrupt.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-14: SECPPCINTSTAT bit descriptions

Bits	Name	Description	Type	Reset
[31:24]	-	Reserved	RAZ/WI	`0x000`
[23:20]	SMAINPPCEXP_STATUS	Interrupt Status of Expansion Peripheral Protection Controller on the Main Interconnect. Each bit *i* shows the status of PPC connected to the input signal SMAINPPCEXPSTATUS[i].	RO	`0x0`
[19:17]	-	Reserved	RAZ/WI	`0x0`
[16]	SMAINPPC0_STATUS	Interrupt Status of Peripheral Protection Controller 0 on the Main Interconnect (MIPPC0). The boot ROM is protected by this PPC. For more information see Read-only memory.	RO	`0x0`
[15:8]	-	Reserved	RAZ/WI	`0x000`
[7:4]	SPERIPHPPCEXP_STATUS	Interrupt Status of Expansion Peripheral Protection Controller on the Peripheral Interconnect. Each bit *i* shows the status of PPC connected to the input signal SPERIPHPPCEXPSTATUS[i].	RO	`0x0`
[3:2]	-	Reserved	RAZ/WI	`0x0`
[1]	SPERIPHPPC1_STATUS	Interrupt Status of Peripheral Protection Controller 1 on the Peripheral Interconnect within the System (PIPPC1).	RO	`0x0`
[0]	SPERIPHPPC0_STATUS	Interrupt Status of Peripheral Protection Controller 0 on the Peripheral Interconnect within the System (PIPPC0).	RO	`0x0`

8.5.2.7 SECPPCINTCLR

When access violations occur on any Peripheral Protection Controller, a level interrupt is raised through a combined interrupt that is then sent to the CPUs. The PPC Secure PPC Interrupt Clear Register allows software to clear the interrupt.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-15: SECPPCINTCLR bit descriptions

Bits	Name	Description	Type	Reset
[31:24]	-	Reserved	RAZ/WI	`0x000`
[23:20]	SMAINPPCEXP_CLR	Interrupt Clear of Expansion Peripheral Protection Controller on the Main Interconnect. Each bit *i*, when set to HIGH clears the interrupt status of the PPC connected to SMAINPPCEXPSTATUS[i] and SMAINPPCEXPCLEAR[i].	RAZ/W1C	`0x0`
[19:17]	-	Reserved	RAZ/WI	`0x0`
[16]	SMAINPPC0_CLR	Interrupt Clear of Peripheral Protection Controller 0 on the Main Interconnect (MIPPC0). Write '1' to clear SMAINPPC0_STATUS. The boot ROM is protected by this PPC. For more information see Read-only memory.	RAZ/W1C	`0x0`
[15:8]	-	Reserved	RAZ/WI	`0x000`
[7:4]	SPERIPHPPCEXP_CLR	Interrupt Clear of Expansion Peripheral Protection Controller on the Peripheral Interconnect. Each bit SPERIPHPPCEXP_CLR[i], when set to HIGH clears the interrupt status of the PPC connected to SPERIPHPPCEXPSTATUS[i] and SPERIHPPCEXPCLEAR[i].	RAZ/W1C	`0x0`
[3:2]	-	Reserved	RAZ/WI	`0x0`
[1]	SPERIPHPPC1_CLR	Interrupt Clear of Peripheral Protection Controller 1 on the Peripheral Interconnect within the System (PIPPC1).	RAZ/W1C	`0x0`
[0]	SPERIPHPPC0_CLR	Interrupt Clear of Peripheral Protection Controller 0 on the Peripheral Interconnect within the System (PIPPC0).	RAZ/W1C	`0x0`

8.5.2.8 SECPPCINTEN

When access violations occur on any Peripheral Protection Controller, a level interrupt is raised through a combined interrupt that is then sent to the CPUs. The PPC Secure PPC Interrupt Status Enable Register allows software to enable or disable (Mask) the interrupt.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-16: SECPPCINTEN bit descriptions

Bits	Name	Description	Type	Reset
[31:24]	-	Reserved	RAZ/ WI	`0x000`
[23:20]	SMAINPPCEXP_EN	Interrupt Enable of Expansion Peripheral Protection Controller on the Main Interconnect. Write '1' to bit *i* enable interrupt from SMAINPPCEXP_STATUS[i].	RW	`0x0`
[19:17]	-	Reserved	RAZ/ WI	`0x0`
[16]	SMAINPPC0_EN	Interrupt Enable of Peripheral Protection Controller 0 on the Main Interconnect (MIPPC0). Write '1' to enable interrupt from SMAINPPC0_STATUS. The boot ROM is protected by this PPC. For more information see Read-only memory.	RW	`0x0`
[15:8]	-	Reserved	RAZ/ WI	`0x000`
[7:4]	SPERIPHPPCEXP_EN	Interrupt Enable of Expansion Peripheral Protection Controller on the Peripheral Interconnect. Write '1' to bit *i* to enable interrupt from SPERIPHPPCEXP_STATUS [i].	RW	`0x0`
[3:2]	-	Reserved	RAZ/ WI	`0x0`
[1]	SPERIPHPPC1_EN	Interrupt Enable of Peripheral Protection Controller 1 on the Peripheral Interconnect within the System (PIPPC1). Write '1' to enable interrupt from SPERIPHPPC1_STATUS.	RW	`0x0`
[0]	SPERIPHPPC0_EN	Interrupt Enable of Peripheral Protection Controller 0 on the Peripheral Interconnect within the System (PIPPC0). Write '1' to enable interrupt from SPERIPHPPC0_STATUS.	RW	`0x0`

8.5.2.9 SECMSCINTSTAT

When a security violation occurs at any Manager Security Controller (MSC) in the subsystem and in the expansion logic, an interrupt is raised through a combined interrupt to the CPUs. The Secure MSC Interrupt Status Register allows software to determine the source of the interrupt.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-17: SECMSCINTSTAT bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	SMSCEXP_STATUS	Interrupt Status for Expansion MSC. Each bit *i* shows the interrupt status of MSC connected to the input signal SMSCEXPSTATUS[i] The configuration option MSCEXPDIS defines if each bit within this register is actually implemented such that if MSCEXPDIS[i] = 1'b1 then SMSCEXP_STATUS[i] is disabled and **RAZ/WI**.	RO	0x0000
[15:12]	-	Reserved	RAZ/WI	0x0
[11]	SMSCNPU3M1_STATUS	Interrupt status for NPU3 M1 MSC **RAZ/WI** If NUMNPU < 4	RO	0x0
[10]	SMSCNPU3M0_STATUS	Interrupt status for NPU3 M0 MSC **RAZ/WI** If NUMNPU < 4	RO	0x0
[9]	SMSCNPU2M1_STATUS	Interrupt status for NPU2 M1 MSC **RAZ/WI** If NUMNPU < 3	RO	0x0
[8]	SMSCNPU2M0_STATUS	Interrupt status for NPU2 M0 MSC **RAZ/WI** If NUMNPU < 3	RO	0x0
[7]	SMSCNPU1M1_STATUS	Interrupt status for NPU1 M1 MSC **RAZ/WI** If NUMNPU < 2	RO	0x0
[6]	SMSCNPU1M0_STATUS	Interrupt status for NPU1 M0 MSC **RAZ/WI** If NUMNPU < 2	RO	0x0
[5]	SMSCNPU0M1_STATUS	Interrupt status for NPU0 M1 MSC **RAZ/WI** If NUMNPU < 1	RO	0x0
[4]	SMSCNPU0M0_STATUS	Interrupt status for NPU0 M0 MSC **RAZ/WI** If NUMNPU < 1	RO	0x0
[3]	SMSCDMAM1_STATUS	Interrupt status for DMA M1 MSC **RAZ/WI** If NUMDMA < 1	RO	0x0
[2]	SMSCDMAM0_STATUS	Interrupt status for DMA M0 MSC **RAZ/WI** If NUMDMA < 1	RO	0x0
[1:0]	-	Reserved	RAZ/WI	0x0

8.5.2.10 SECMSCINTCLR

When a security violation occurs at any Manager Security Controller (MSC) in the subsystem and in the expansion logic, an interrupt is raised through a combined interrupt to the CPUs. The Secure MSC Interrupt Clear Register allows software to clear the interrupt.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-18: SECMSCINTCLR bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	SMSCEXP_CLR	Interrupt Clear for Expansion MSC. Each bit *i*, when set to HIGH clears the interrupt status of the MSC connected to SMSCEXPSTATUS[i] and SMSCEXPCLEAR[i] The configuration option MSCEXPDIS defines whether each bit within this register is actually implemented such that if MSCEXPDIS[*i*] = 1'b1 then SMSCEXP_CLR[*i*] is disabled and **RAZ/WI**.	RAZ/ W1C	`0x0000`
[15:12]	-	Reserved	RAZ/ WI	`0x0`
[11]	SMSCNPU3M1_CLR	Interrupt clear for NPU3 M1 MSC **RAZ/WI** If NUMNPU < 4	RAZ/ W1C	`0x0`
[10]	SMSCNPU3M0_CLR	Interrupt clear for NPU3 M0 MSC **RAZ/WI** If NUMNPU < 4	RAZ/ W1C	`0x0`
[9]	SMSCNPU2M1_CLR	Interrupt clear for NPU2 M1 MSC **RAZ/WI** If NUMNPU < 3	RAZ/ W1C	`0x0`
[8]	SMSCNPU2M0_CLR	Interrupt clear for NPU2 M0 MSC **RAZ/WI** If NUMNPU < 3	RAZ/ W1C	`0x0`
[7]	SMSCNPU1M1_CLR	Interrupt clear for NPU1 M1 MSC **RAZ/WI** If NUMNPU < 2	RAZ/ W1C	`0x0`

Bits	Name	Description	Type	Reset
[6]	SMSCNPU1M0_CLR	Interrupt clear for NPU1 M0 MSC **RAZ/WI** If NUMNPU < 2	RAZ/ W1C	`0x0`
[5]	SMSCNPU0M1_CLR	Interrupt clear for NPU0 M1 MSC **RAZ/WI** If NUMNPU < 1	RAZ/ W1C	`0x0`
[4]	SMSCNPU0M0_CLR	Interrupt clear for NPU0 M0 MSC **RAZ/WI** If NUMNPU < 1	RAZ/ W1C	`0x0`
[3]	SMSCDMAM1_CLR	Interrupt clear for DMA M1 MSC **RAZ/WI** If NUMDMA < 1	RAZ/ W1C	`0x0`
[2]	SMSCDMAM0_CLR	Interrupt clear for DMA M0 MSC **RAZ/WI** If NUMDMA < 1	RAZ/ W1C	`0x0`
[1:0]	-	Reserved	RAZ/ WI	`0x0`

8.5.2.11 SECMSCINTEN

When a security violation occurs at any Manager Security Controller (MSC) in the subsystem and in the expansion logic, an interrupt is raised through a combined interrupt to the CPUs. The Secure MSC Interrupt Enable Register allows software to enable or disable (mask) the interrupt.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-19: SECMSCINTEN bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	SMSCEXP_EN	Interrupt Enable for Expansion MSC. Each bit *i* enables or disables the input interrupt signal SMSCEXPSTATUS[i]. The parameter MSCEXPDIS defines if each bit within this register is actually implement such that if MSCEXPDIS[*i*] = 1'b1, then SMSCEXP_EN[*i*] is disabled and **RAZ/WI**.	RW	0x0000
[15:12]	-	Reserved	RAZ/WI	0x0
[11]	SMSCNPU3M1_EN	Interrupt enable for NPU3 M1 MSC **RAZ/WI** If NUMNPU < 4	RW	0x0
[10]	SMSCNPU3M0_EN	Interrupt enable for NPU3 M0 MSC **RAZ/WI** If NUMNPU < 4	RW	0x0
[9]	SMSCNPU2M1_EN	Interrupt enable for NPU2 M1 MSC **RAZ/WI** If NUMNPU < 3	RW	0x0
[8]	SMSCNPU2M0_EN	Interrupt enable for NPU2 M0 MSC **RAZ/WI** If NUMNPU < 3	RW	0x0
[7]	SMSCNPU1M1_EN	Interrupt enable for NPU1 M1 MSC **RAZ/WI** If NUMNPU < 2	RW	0x0
[6]	SMSCNPU1M0_EN	Interrupt enable for NPU1 M0 MSC **RAZ/WI** If NUMNPU < 2	RW	0x0
[5]	SMSCNPU0M1_EN	Interrupt enable for NPU0 M1 MSC **RAZ/WI** If NUMNPU < 1	RW	0x0
[4]	SMSCNPU0M0_EN	Interrupt enable for NPU0 M0 MSC **RAZ/WI** If NUMNPU < 1	RW	0x0
[3]	SMSCDMAM1_EN	Interrupt enable for DMA M1 MSC **RAZ/WI** If NUMDMA < 1	RW	0x0
[2]	SMSCDMAM0_EN	Interrupt enable for DMA M0 MSC **RAZ/WI** If NUMDMA < 1	RW	0x0
[1:0]	-	Reserved	RAZ/WI	0x0

8.5.2.12 BRGINTSTAT

The CRSAS Ma2 and its expansion logic can contain bus bridges which 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 subordinate interfaces before any potential error response is received for the write access on their manager interfaces. When this occurs, these bridges can raise a combined interrupt. The Bridge Buffer Error Interrupt Status Register allows software to determine source of the interrupt.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-20: BRGINTSTAT bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	BRGEXP_STATUS	Interrupt Status for Expansion Bridge Buffer Error Interrupts. Each bit *i* shows the interrupt status of the bridge connected to the input signal BRGEXPSTATUS[i]. The configuration option BRGEXPDIS defines if each bit within this register is actually implemented such that if BRGEXPDIS[*i*] = 1'b1, then BRGEXP_STATUS[*i*] is disabled and **RAZ/WI**.	RO	`0x0000`
[15:0]	-	Reserved	RAZ/WI	`0x0000`

8.5.2.13 BRGINTCLR

The CRSAS Ma2 and its expansion logic can contain bus bridges which 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 subordinate interfaces before any potential error response is received for the write access on their manager interfaces. When this occurs, these bridges can raise a combined interrupt. The Bridge Buffer Error Interrupt Clear Register allows software to clear the interrupt.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-21: BRGINTCLR bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	BRGEXP_CLR	Interrupt Clear of Expansion Bridge Buffer Error Interrupts. Each bit *i* when set to HIGH clears the interrupt status of the bridge connected to BRGEXPSTATUS[i] and BRGEXPCLEAR[i]. The configuration option BRGEXPDIS defines if each bit within this register is actually implemented such that if BRGEXPDIS[*i*] = 1'b1 then BRGEXP_CLR[*i*] is disabled and **RAZ/WI**.	RAZ/ W1C	`0x0000`
[15:0]	-	Reserved	RAZ/ WI	`0x0000`

8.5.2.14 BRGINTEN

The CRSAS Ma2 and its expansion logic can contain bus bridges which 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 subordinate interfaces before any potential error response is received for the write access on their manager interfaces. When this occurs, these bridges can raise a combined interrupt. The Bridge Buffer Error Interrupt Enable Register allows software to enable or disable (mask) the interrupt.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-22: BRGINTEN bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	BRGEXP_EN	Interrupt Enable of Expansion Bridge Buffer Error Interrupts. Each bit *i* enables the interrupt of the bridge connected to BRGEXPSTATUS[i] The configuration option BRGEXPDIS defines if each bit within this register is actually implemented such that if BRGEXPDIS[*i*] = 1'b1, then BRGEXP_EN[*i*] is disabled and **RAZ/WI**.	RW	`0x0000`
[15:0]	-	Reserved	RAZ/ WI	`0x0000`

8.5.2.15 MAINNSPPC0

The Main Interconnect Non-secure Access Peripheral Protection Controller Register allows software to configure if each peripheral on the Main Interconnect that it controls through a 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'
  Allow Secure access only

'1'
  Allow Non-secure access only

The CRSAS Ma2 currently do not have any interfaces on the Main Interconnect that need security configuration support of the PPC. Therefore, this register is reserved.

Configurations

This register is available in all configurations.

Attributes

Width
  32-bit

Power domain
  PD_SYS

Reset domain
  nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-23: MAINNSPPC0 bit descriptions

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RAZ/WI	`0x0000_0000`

8.5.2.16 MAINNSPPCEXP{0-3}

The Main Interconnect Non-secure Access Subordinate Peripheral Protection Controller Expansion register 0, 1, 2, and 3 allow software to configure the security level of each Main Interconnect peripheral that resides in the subsystem expansion outside the subsystem. These registers can be used to control the PPCs - outside the subsystem - that are protecting the Main Interconnect peripherals connected to the Main Interconnect through the Subordinate Main Expansion interfaces.

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

‘0’
: Allow Secure access only

‘1’
: Allow Non-secure access only

These controls directly control the expansion signals on the Security Control Expansion interface. All four registers are similar and each register *x*, where *x* is from 0 to 3, is defined as seen in the Bit descriptions table.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width
: 32-bit

Power domain
: PD_SYS

Reset domain
: nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-24: MAINNSPPCEXP<x> bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RAZ/WI	`0x0000`
[15:0]	MAINNSPPCEXP<x>	Expansion <x> Non-Secure Access Main Interconnect Subordinate Peripheral Protection Control. Each bit *i* drives the output signal MAINNSPPCEXP<x>[i]. The configuration option MAINPPCEXP<x>DIS defines if each bit within this register is actually implemented such that if MAINPPCEXP<x>DIS[*i*] = 1'b1, then MAINNSPPCEXP<x>[*i*] is disabled and **RAZ/WI**.	RW	`0x0000`

8.5.2.17 PERIPHNSPPC0

The Peripheral Interconnect Non-secure Access Peripheral Protection Controller Registers allows software to configure if each peripheral on the Peripheral Interconnect that it controls through a 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:

'1'
    Allow Non-secure access only

'0'
    Allow Secure access only

See also PERIPHNSPPC1.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width
    32-bit

Power domain
    PD_SYS

Reset domain
    nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-25: PERIPHNSPPC0 bit descriptions

Bits	Name	Description	Type	Reset
[31:9]	-	Reserved	RAZ/WI	0x000_000
[8]	NS_SDC	Access Security for SDC-600. If SDC-600 does not exist, this field is reserved and **RAZ/WI**.	RW	0x0
[7]	NS_SYSDSS	Access Security for interconnect access to Debug System. When HASCSS = 0, this field is reserved and **RAZ/WI**.	RW	0x0
[6]	-	Reserved	RAZ/WI	0x0
[5]	NS_TIMER3	Access Security for TIMER3	RW	0x0
[4]	NS_MHU1	Access Security for MHU 1. When NUMCPU = 0, this field is reserved and **RAZ/WI**.	RW	0x0
[3]	NS_MHU0	Access Security for MHU 0. When NUMCPU = 0, this field is reserved and **RAZ/WI**.	RW	0x0
[2]	NS_TIMER2	Access Security for TIMER2	RW	0x0
[1]	NS_TIMER1	Access Security for TIMER1	RW	0x0
[0]	NS_TIMER0	Access Security for TIMER0	RW	0x0

Note

Access to several peripherals are filtered by PIPPC0 even though its security setting is fixed and is not represented in the preceding table. Because of this, if such peripheral is accessed using the wrong security it also result in interrupts being raised for PIPPC0. Full list of peripherals protected by PIPPC0 are listed in Peripheral Protection Controllers.

8.5.2.18 PERIPHNSPPC1

The Peripheral Interconnect Non-secure Access Peripheral Protection Controller Registers allow software to configure if each peripheral on the Peripheral Interconnect that it controls through a 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:

‘1’

Allow Non-secure access only

‘0’

Allow Secure access only

See also PERIPHNSPPC0.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-26: PERIPHNSPPC1 bit descriptions

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved	RAZ/WI	`0x0000_0000`
[0]	NS_SLOWCLK_TIMER	Access Security for SLOWCLK_TIMER	RW	`0x0`

Note

Access to several peripherals filtered by PIPPC1 though its security setting is fixed and is not represented in the preceding table. Because of this, if such peripheral is accessed using the wrong security it also result in interrupts being raised for PIPPC1. Full list of peripherals protected by PIPPC1 are listed in Peripheral Protection Controllers.

8.5.2.19 NPUSPPORSL

The NPU power on reset Security Level Control Register allows software to configure if each NPU resets to Secure or Non-secure state. The value of this register is only sampled by the NPU, when the NPU is released from reset.

The default reset value of this register is controlled by NPU<m>PORSLRST:

‘1’

Reset to Non-secure state

‘0’

Reset to Secure state

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-27: NPUSPPORSL bit descriptions

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RAZ/WI	`0x000_000`
[3]	SP_NPU3PORSL	Configures the security level strap value for NPU3 This field does not exist, is reserved and **RAZ/WI** when NUMNPU < 4.	RW	NPU3PORSLRST
[2]	SP_NPU2PORSL	Configures the security level strap value for NPU2 This field does not exist, is reserved and **RAZ/WI** when NUMNPU < 3.	RW	NPU2PORSLRST
[1]	SP_NPU1PORSL	Configures the security level strap value for NPU1 This field does not exist, is reserved and **RAZ/WI** when NUMNPU < 2.	RW	NPU1PORSLRST
[0]	SP_NPU0PORSL	Configures the security level strap value for NPU0 This field does not exist, is reserved and **RAZ/WI** when NUMNPU < 1.	RW	NPU0PORSLRST

8.5.2.20 PERIPHNSPPCEXP{0-3}

The Peripheral Interconnect Non-secure Access Subordinate Peripheral Protection Controller Expansion registers 0, 1, 2, and 3 allow software to configure the security level of each Peripheral Interconnect peripheral that resides in the expansion logic outside the subsystem.

These registers can be used to control the PPCs - outside the subsystem- that are protecting the Periperheral Interconnect peripherals connected to the Peripheral Interconnect through the Subordinate Peripheral Expansion interfaces.

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

'1'

Allow Non-secure access only

'0'

Allow Secure access only

These controls directly control the expansion signals on the Security Control Expansion interface. All four registers are similar and each register *x*, where *x* is from 0 to 3, is defined as seen in the Bit descriptions table.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-28: PERIPHNSPPCEXP<x> bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RAZ/WI	`0x0000`
[15:0]	PERIPHNSPPCEXP<x>	Expansion <x> Non-Secure Access Peripheral Interconnect Subordinate Peripheral Protection Control. Each bit *x* drives the output signal PERIPHNSPPCEXP<x>[i]. The configuration option PERIPHPPCEXP<x>DIS defines if each bit within this register is actually implemented such that if PERIPHPPCEX<x>PDIS[*i*] = 1'b1, then PERIPHNSPPCEXP<x>[*i*] is disabled, reads as zeros and any writes to it is ignored.	RW	`0x0000`

8.5.2.21 MAINSPPPC0

The Secure Unprivileged Access Main Interconnect Subordinate Peripheral Protection Controller Register allows software to configure if each peripheral on the Main Interconnect that it controls through a PPC is only allowed Secure Privileged Access or is also allowed Secure Unprivileged access.

Each field defines this for an associated peripheral, by the following settings:

'0'

Allow Secure privileged access only

'1'

Allow Secure unprivileged and privileged access

The CRSAS Ma2 currently does not have any Main Interconnect subordinate interfaces that need Secure Unprivileged Access configuration support of the PPC. Therefore, this register is reserved.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-29: MAINSPPPC0 bit descriptions

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RAZ/WI	`0x0000_0000`

8.5.2.22 MAINSPPPCEXP{0-3}

The Expansion Secure Privileged Access Main Interconnect Subordinate Peripheral Protection Controller Registers 0, 1, 2 and 3 allow software to configure each Main Interconnect peripheral that it controls through each PPC, that resides in the expansion logic outside the subsystem, is Secure privileged Access only or is also allowed Secure Unprivileged access.

Each field defines this for an associated peripheral, by the following settings:

‘0’

Allow Secure privileged access only

‘1’

Allow Secure unprivileged and privileged access

These settings directly control the expansion signals on the Security Control Expansion interface. All four registers are similar and each register x, where x is from 0 to 3, is defined as seen in the Bit descriptions table.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-30: MAINSPPPCEXP<x> bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RAZ/WI	0x0000
[15:0]	MAINSPPPCEXP<x>	Expansion <x> Secure Privileged Access Main Interconnect Subordinate Peripheral Protection Control. Each bit *n* drives the output signal MAINPPPCEXP<x>[i] if MAINNSPPCEXP<x>. MAINNSPPCEXP<x>[i] is also LOW, where *x* is 0 to 3. The configuration point MAINPPCEXP<x>DIS defines if each bit within this register is actually implemented such that if MAINPPCEXP<x>DIS[*i*] = 1'b1, then MAINSPPPCEXP<x>[*i*] is disabled and **RAZ/WI**.	RW	0x0000

8.5.2.23 PERIPHSPPPC0

The Secure Unprivileged Access Peripheral Interconnect Subordinate Peripheral Protection Controller Register allows software to configure if each Peripheral Interconnect peripheral that it controls through a PPC is only allowed Secure privileged access or is also allowed Secure unprivileged access.

Each field defines this for an associated peripheral, by the following settings:

'0'

Allow Secure privileged access only

'1'

Allow Secure Unprivileged and privileged access

The CRSAS Ma2 has two such registers, see also PERIPHSPPPC1.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-31: PERIPHSPPPC0 bit descriptions

Bits	Name	Description	Type	Reset
[31:9]	-	Reserved	RAZ/WI	`0x000_000`
[8]	SP_SDC	Secure privileged setting for SDC-600. If SDC-600 does not exist, this field is reserved and **RAZ/WI**.	RW	`0x0`
[7]	SP_SYSDSS	Secure privileged setting for interconnect access to Debug System. When HASCSS = 0, this field is reserved and **RAZ/WI**.	RW	`0x0`
[6]	SP_WATCHDOG_REF	Secure System Watchdog Refresh Frame.	RW	`0x0`
[5]	SP_TIMER3	Secure privileged setting for TIMER3.	RW	`0x0`
[4]	SP_MHU1	Secure privileged setting for MHU 1. When NUMCPU = 0, this field is reserved and **RAZ/WI**.	RW	`0x0`
[3]	SP_MHU0	Secure privileged setting for MHU 0. When NUMCPU = 0, this field is reserved and **RAZ/WI**.	RW	`0x0`
[2]	SP_TIMER2	Secure privileged setting for TIMER2.	RW	`0x0`
[1]	SP_TIMER1	Secure privileged setting for TIMER1.	RW	`0x0`
[0]	SP_TIMER0	Secure privileged setting for TIMER0.	RW	`0x0`

8.5.2.24 PERIPHSPPPC1

The Secure Unprivileged Access Peripheral Interconnect Subordinate Peripheral Protection Controller Register allows software to configure if each Peripheral Interconnect peripheral that it controls through a PPC is only allowed Secure privileged access or is allowed Secure unprivileged access also.

Each field defines this for an associated peripheral, by the following settings:

‘0’

Allow Secure privileged access only

‘1’

Allow Secure Unprivileged and privileged access

The CRSAS Ma2 has two such registers, see also PERIPHSPPPC0.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-32: PERIPHSPPPC1 bit descriptions

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RAZ/WI	0x0
[3]	SP_LCM	Secure privileged setting for Lifecycle Manager subordinate interface When LCM_KMU_SAM_PRESENT = 0 this field is Reserved and **RAZ/WI**.	RW	0x0
[2]	SP_SAM	Secure privileged setting for Security Alarm Manager subordinate interface When LCM_KMU_SAM_PRESENT = 0 this field is Reserved and **RAZ/WI**.	RW	0x0
[1]	SP_KMU	Secure privileged setting for Key Management Unit subordinate APB3 programming port and for AHB5 Manager interface both When LCM_KMU_SAM_PRESENT = 0 this field is Reserved and **RAZ/WI**.	RW	0x0
[0]	SP_SLOWCLK_TIMER	Secure privileged setting for SLOWCLK_TIMER.	RW	0x0

8.5.2.25 NPUSPPORPL

The NPU power on reset Secure Access Privilege Level Control Register allows software to configure if each NPU resets to privileged or unprivileged state. The value of this register is only sampled by the NPU, when the NPU is released from reset and NPUSPPORSL.SP_NPU<m>PORSL is Secure State.

The default reset value of this register is controlled by NPU<m>PORPLRST:

‘1’

Reset to privileged state

‘0’

Reset to unprivileged state

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-33: NPUSPPORPL bit descriptions

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RAZ/WI	`0x0000_000`
[3]	SP_NPU3PORPL	Configures the Secure access privilege level strap value for NPU3 This field does not exist, is reserved and **RAZ/WI** when NUMNPU < 4.	RW	NPU3PORPLRST
[2]	SP_NPU2PORPL	Configures the Secure access privilege level strap value for NPU2 This field does not exist, is reserved and **RAZ/WI** when NUMNPU < 3.	RW	NPU2PORPLRST
[1]	SP_NPU1PORPL	Configures the Secure access privilege level strap value for NPU1 This field does not exist, is reserved and **RAZ/WI** when NUMNPU < 2.	RW	NPU1PORPLRST
[0]	SP_NPU0PORPL	Configures the Secure access privilege level strap value for NPU0 This field does not exist, is reserved and **RAZ/WI** when NUMNPU < 1.	RW	NPU0PORPLRST

8.5.2.26 PERIPHSPPPCEXP{0-3}

The Expansion Secure Privileged Access Peripheral Interconnect Subordinate Peripheral Protection Controller registers 0, 1, 2 and 3 allow software to configure each Peripheral Interconnect peripheral that it controls through each PPC, that resides in the expansion logic outside the subsystem, is allowed Secure privileged Access only or is allowed Secure Unprivileged access also.

Each field defines this for an associated peripheral, by the following settings:

'0'

Allow Secure privileged access only.

'1'

Allow Secure unprivileged and privileged access.

These directly control the expansion signals on the Security Control Expansion interface. All four registers are similar and each register x, where x is from 0 to 3, is defined as seen in the Bit descriptions table.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-34: PERIPHSPPPCEXP<x> bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RAZ/WI	`0x0000`
[15:0]	PERIPHSPPPCEXP<x>	Expansion <x> Secure Privileged Access Peripheral Interconnect Subordinate Peripheral Protection Control. Each bit *i* drives the output signal PERIPHPPPCEXP<x>[i] if PERIPHNSPPCEXP<x>. PERIPHNSPPCEXP<x>[i] is also LOW, where *x* is 0 to 3. The configuration option PERIPHPPCEXP<*x*>DIS defines if each bit within this register is actually implemented such that if PERIPHPPCEXP<x>DIS[*i*] = 1'b1, then PERIPHSPPPCEXP<x>[*i*] is disabled and **RAZ/WI**.	RW	`0x0000`

8.5.2.27 NSMSCEXP

The Non-secure Expansion Manager Security Controller Register allows software to configure if each manager that is located behind each MSC in the subsystem expansion is a Secure or Non-secure device.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-35: NSMSCEXP bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	NS_MSCEXP	Expansion MSC Non-secure Configuration. Each bit *i* controls the Non-secure configuration of each MSC and drives the signal NSMSCEXP[i], set HIGH to define a Manager as Non-secure, or LOW for Secure. The parameter MSCEXPDIS defines if each bit within this register is actually implemented such that if MSCEXPDIS[*i*] = 1'b1 then NS_MSCEXP[*i*] is disabled and **RAO/WI**. Resets to NSMSCEXPRST.	RW	NSMSCEXPRST
[15:0]	-	Reserved	RAZ/WI	`0x0000`

8.5.3 Non-secure Access Configuration Register block

The Non-secure Access Configuration Register block implements program visible states that allow software to control various security gating units within the design. This register block base address is `0x4008_0000`. These registers are Non-secure privileged access only and support 32-bit RW accesses. For write access to these registers, only 32-bit writes are supported. Any byte and halfword writes are ignored.

All registers reside in the PD_SYS power domain and are reset by nWARMRESETSYS.

The following table lists the registers within this unit. Details of each register are described in the following subsections.

Table 8-36: Non-secure Access Configuration Register block register map

Offset	Name	Type	Reset	Description
`0x000-0x08C`		RAZ/WI	`0x0000_0000`	Reserved
`0x090`	MAINNSPPPC0	RW	`0x0000_0000`	Non-secure Unprivileged Access Peripheral Protection Control 0 on Main Interconnect, see MAINNSPPPC0.
`0x094-0x09C`		RAZ/WI	`0x0000_0000`	Reserved
`0x0A0`	MAINNSPPPCEXP0	RW	`0x0000_0000`	Expansion 0 Non-secure Unprivileged Access Peripheral Protection Control on Main Interconnect, see MAINNSPPPCEXP{0-3}.
`0x0A4`	MAINNSPPPCEXP1	RW	`0x0000_0000`	Expansion 1 Non-secure Unprivileged Access Peripheral Protection Control on Main Interconnect, see MAINNSPPPCEXP{0-3}.

Offset	Name	Type	Reset	Description
0x0A8	MAINNSPPPCEXP2	RW	0x0000_0000	Expansion 2 Non-secure Unprivileged Access Peripheral Protection Control on Main Interconnect, see MAINNSPPPCEXP{0-3}.
0x0AC	MAINNSPPPCEXP3	RW	0x0000_0000	Expansion 3 Non-secure Unprivileged Access Peripheral Protection Control on Main Interconnect, see MAINNSPPPCEXP{0-3}.
0x0B0	PERIPHNSPPPC0	RW	0x0000_0000	Non-secure Unprivileged Access Peripheral Protection Control 0 on Peripheral Interconnect, see PERIPHNSPPPC0.
0x0B4	PERIPHNSPPPC1	RW	0x0000_0000	Non-secure Unprivileged Access Peripheral Protection Control 1 on Peripheral Interconnect, see PERIPHNSPPPC1.
0x0B8	NPUNSPORPL	RW	NPU<m>PORPLRST	NPU power on reset Non-secure access privilege level control register, Each Field Controls the PORPL inputs of the associated NPU see NPUNSPORPL.
0x0BC		RAZ/WI	0x0000_0000	Reserved
0x0C0	PERIPHNSPPPCEXP0	RW	0x0000_0000	Expansion 0 Non-secure Unprivileged Access Peripheral Protection Control on Peripheral Interconnect, see PERIPHNSPPPCEXP{0-3}.
0x0C4	PERIPHNSPPPCEXP1	RW	0x0000_0000	Expansion 1 Non-secure Unprivileged Access Peripheral Protection Control on Peripheral Interconnect, see PERIPHNSPPPCEXP{0-3}.
0x0C8	PERIPHNSPPPCEXP2	RW	0x0000_0000	Expansion 2 Non-secure Unprivileged Access Peripheral Protection Control on Peripheral Interconnect, see PERIPHNSPPPCEXP{0-3}.
0x0CC	PERIPHNSPPPCEXP3	RW	0x0000_0000	Expansion 3 Non-secure Unprivileged Access Peripheral Protection Control on Peripheral Interconnect, see PERIPHNSPPPCEXP{0-3}.
0x0D0 -0xFCC		RAZ/WI	0x0000_0000	Reserved
0xFD0	PIDR4	RO	0x0000_0004	Peripheral ID 4
0xFD4 -0xFDC		RAZ/WI	0x0000_0000	Reserved
0xFE0	PIDR0	RO	0x0000_0053	Peripheral ID 0
0xFE4	PIDR1	RO	0x0000_00B8	Peripheral ID 1
0xFE8	PIDR2	RO	0x0000_003B	Peripheral ID 2
0xFEC	PIDR3	RO	0x0000_0000	Peripheral ID 3
0xFF0	CIDR0	RO	0x0000_000D	Component ID 0
0xFF4	CIDR1	RO	0x0000_00F0	Component ID 1
0xFF8	CIDR2	RO	0x0000_0005	Component ID 2
0xFFC	CIDR3	RO	0x0000_00B1	Component ID 3

8.5.3.1 MAINNSPPPC0

Non-secure Unprivileged Access Main Interconnect Subordinate Peripheral Protection Controller Register allows software to configure if each peripheral on the Main Interconnect that it controls through a PPC is only allowed Non-secure Privileged Access or is also allowed Non-secure Unprivileged access.

Each field defines this for an associated peripheral, by the following settings:

'0'

Allow Non-secure privileged access only

'1'

Allow Non-secure unprivileged and privileged access

The CRSAS Ma2 currently does not have any Main Interconnect subordinate interfaces that need Non-secure Unprivileged Access configuration support of the PPC. Therefore, this register is reserved.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Non-secure privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-37: MAINNSPPPC0 bit descriptions

Bits	Name	Description	Type	Reset
[31:0]	-	Reserved	RAZ/WI	`0x0000_0000`

8.5.3.2 MAINNSPPPCEXP{0-3}

The Expansion Non-secure Unprivileged Access Main Interconnect Subordinate Peripheral Protection Controller registers 0, 1, 2 and 3 allow software to configure each Main Interconnect peripheral that it controls through each PPC, that resides in the expansion logic outside the subsystem, is only Non-secure privileged Access or is also allowed Non-secure Unprivileged access.

Each field defines this for an associated peripheral, by the following settings:

'0'

Allow Non-secure privileged access only

'1'

Allow Non-secure unprivileged and privileged access

These settings directly control the expansion signals on the Security Control Expansion interface. All four register are similar and each register, x where x is from 0 to 3 is as seen in the Bit descriptions table.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Non-secure privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-38: MAINNSPPPCEXP<x> bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RAZ/ WI	`0x0000`
[15:0]	MAINNSPPPCEXP <x>	Expansion <x> Non-secure Privileged Access Main Interconnect Subordinate Peripheral Protection Control. Each bit *i* drives the output signal MAINPPPCEXP<x>[i] if MAINSPPPCEXP<x>. MAINNSPPCEXP<x>[i] is also HIGH, where *x* is 0 to 3. The configuration point MAINPPCEXP<x>DIS defines if each bit within this register is actually implemented such that if MAINPPCEXP<x>DIS[i] = 1'b1 then MAINNSPPPCEXP<x>[i] is disabled and **RAZ/WI**.	RW	`0x0000`

8.5.3.3 PERIPHNSPPPC0

Non-secure Unprivileged Access Peripheral Interconnect Subordinate Peripheral Protection Controller Register allows software to configure if each Peripheral Interconnect peripheral that it controls through a PPC is only Non-secure privileged access or is also allowed Non-secure unprivileged access.

Each field defines this for an associated peripheral, by the following settings:

'0'

Allow Non-secure privileged access only

'1'

Allow Non-secure unprivileged and privileged access

The CRSAS Ma2 has two such registers, see also PERIPHNSPPPC1.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Non-secure privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-39: PERIPHNSPPPC0 bit descriptions

Bits	Name	Description	Type	Reset
[31:9]	-	Reserved	RAZ/ WI	`0x0000_00`
[8]	NSP_SDC	Non-secure privileged setting for SDC-600. If SDC-600 does not exist, this field is reserved and **RAZ/WI**.	RW	`0x0`
[7]	NSP_SYSDSS	Non-secure privileged setting for Debug System. When HASCSS = 0, this field is reserved and **RAZ/WI**.	RW	`0x0`
[6]	NSP_WATCHDOG_REF	Secure System Watchdog Refresh Frame.	RW	`0x0`
[5]	NSP_TIMER3	Non-secure privileged setting for TIMER3.	RW	`0x0`
[4]	NSP_MHU1	Non-secure privileged setting for MHU 1. When NUMCPU = 0, this field is reserved and **RAZ/WI**.	RW	`0x0`
[3]	NSP_MHU0	Non-secure privileged setting for MHU 0. When NUMCPU = 0, this field is reserved and **RAZ/WI**.	RW	`0x0`
[2]	NSP_TIMER2	Non-secure privileged setting for TIMER2.	RW	`0x0`
[1]	NSP_TIMER1	Non-secure privileged setting for TIMER1.	RW	`0x0`
[0]	NSP_TIMER0	Non-secure privileged setting for TIMER0.	RW	`0x0`

8.5.3.4 PERIPHNSPPPC1

Non-secure Unprivileged Access Peripheral Interconnect Subordinate Peripheral Protection Controller Register allows software to configure if each Peripheral Interconnect peripheral that it controls through a PPC is only allowed Non-secure privileged access or is also allowed Non-secure unprivileged access.

Each field defines this for an associated peripheral, by the following settings:

‘0’

Allow Non-secure privileged access only

‘1’

Allow Non-secure unprivileged and privileged access

The CRSAS Ma2 has two such registers, see also PERIPHNSPPPC0.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Non-secure privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-40: PERIPHNSPPPC1 bit descriptions

Bits	Name	Description	Type	Reset
[31:1]	-	Reserved	RAZ/WI	`0x0000_0000`
[0]	NSP_SLOWCLK_TIMER	Non-secure privileged setting for SLOWCLK_TIMER.	RW	`0x0`

8.5.3.5 NPUNSPORPL

The NPU power on reset Non-secure access privilege level control registers allow software to configure if each NPU resets to privileged or unprivileged state.

The value of this register is only sampled by the NPU, when the NPU is released from reset and NPUSPPORSL.SP_NPU<m>PORSL is Non-secure State.

The default reset value of this register is controlled by NPU<m>PORPLRST, and it can take the following values:

'1'

Reset to privileged state

'0'

Reset to unprivileged state

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Non-secure privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-41: NPUNSPORPL bit descriptions

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RAZ/WI	`0x0000_000`
[3]	NS_NPU3PORPL	Configures the Non-secure access privilege level strap value for NPU3. When NUMNPU < 4, This field is reserved and RAZ/WI	RW	NPU3PORPLRST
[2]	NS_NPU2PORPL	Configures the Non-secure access privilege level strap value for NPU2. When NUMNPU < 3, This field is reserved and RAZ/WI	RW	NPU2PORPLRST
[1]	NS_NPU1PORPL	Configures the Non-secure access privilege level strap value for NPU1. When NUMNPU < 2, This field is reserved and RAZ/WI	RW	NPU1PORPLRST
[0]	NS_NPU0PORPL	Configures the Non-secure access privilege level strap value for NPU0. When NUMNPU < 1, This field is reserved and RAZ/WI	RW	NPU0PORPLRST

8.5.3.6 PERIPHNSPPPCEXP{0-3}

The Expansion Non-secure Unprivileged Access Peripheral Interconnect Subordinate Peripheral Protection Controller registers 0, 1, 2, and 3 allow software to configure peripheral interconnects. Each Peripheral Interconnect that it controls through each PPC, that resides in the expansion logic outside the subsystem, is only allowed Non-secure privileged Access or is also allowed Non-secure unprivileged access.

Each field defines this for an associated peripheral, by the following settings:

'0'

Allow Non-secure privileged access only.

'1'

Allow Non-secure unprivileged and privileged access.

These directly controls the expansion signals on the Security Control Expansion interface. All four registers are similar and each register x, where x is from 0 to 3, is defined as seen in the Bit descriptions table.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

PD_SYS

Reset domain

nWARMRESETSYS

Usage constraints

This register is Non-secure privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-42: PERIPHNSPPPCEXP<x> bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RAZ/WI	`0x0000`
[15:0]	PERIPHNSPPPCEXP<x>	Expansion <x> Non-secure Privileged Access Peripheral Interconnect Subordinate Peripheral Protection Control. Each bit *i* drives the output signal PERIPHPPPCEXP<x>[i] if PERIPHNSPPCEXP<x>. PERIPHNSPPCEXP<x>[i] is also HIGH, where *x* is 0 to 3 and *i* is 0 to 15. The configuration option PERIPHPPCEXP<x>DIS defines if each bit within this register is actually implemented such that if PERIPHPPCEXP<x>DIS[i] = 1'b1, where *x* is 0 to 3 and i is 0 to 15, then PERIPHNSPPPCEXP<x>[i] is disabled and **RAZ/WI**.	RW	`0x0000`

8.5.4 Timestamp-based timers registers

The CRSAS Ma2 implements four timestamp-based timers in the system, TIMER<x> where x is 0 to 3. All timers are mapped to the Secure or Non-secure world through PIPPC0, which also controls accessibility of unprivileged accesses.

See Secure access configuration register block.

All timestamp timers, except for Timer 3, reside in the PD_SYS power domain and are reset by nWARMRESETSYS, while the Timer 3 resides in the PD_AON power domain and is reset by nWARMRESETAON.

For more information, see System timer components appendix.

8.5.5 Timestamp-based watchdogs registers

The CRSAS Ma2 implements two timestamp-based watchdogs in the system. All reside in the PD_SYS power domain and are reset by nWARMRESETSYS. One watchdog timer is Secure access only, while another is Non-secure. Each Watchdog Timer implements two register frames, a Control Frame and a Refresh Frame. The Control Frame is always fixed privileged while the Refresh Frame accessibility to unprivileged access is configurable and controlled by PIPPC0.

See PERIPHSPPPC0 and PERIPHNSPPPC0.

For more information, see System Watchdog overview.

8.5.6 Secure Debug Channel registers

The CRSAS Ma2 supports an optional SDC-600 Internal APBCOM, which if it exist, is expected to connect to an external SDC-600 APBCOM in debug expansion. The SDC-600 Internal APBCOM are mapped to the Secure or Non-secure world through PIPPC0, which also controls accessibility of unprivileged accesses.

See Secure access configuration register block.

Note

The CRSAS Ma2 does not support the use of a SDC-600 external APBCOM's REMRR and REMRA remote reboot request handshake interface signals to request for a system reset.

The SDC-600 Internal APBCOM resides in the PD_SYS power domain and is reset by nWARMRESETSYS.

For more information on SDC-600, see *Arm® CoreSight™ SDC-600 Secure Debug Channel Technical Reference Manual*.

8.5.7 DMA registers

The CRSAS Ma2 implements up to one DMA-350.

The DMA handles security and privilege checking of accesses to DMA registers. The DMA resides in the PD_SYS power domain and is reset by nWARMRESETSYS.

For more information about the DMA software interface, see *Arm® CoreLink™ DMA-350 Controller Technical Reference Manual*.

8.5.8 NPU<m> registers

The CRSAS Ma2 implements up to four Ethos-U55 or Ethos-U65 NPUs. The PIPPC0 handles security and privilege checking of accesses to NPU registers.

All NPUs reside in the PD_NPU<*m*> power domain and are reset by nWARMRESETNPU<m>.

For more information about the NPU software interface, see *Arm® Ethos™-U55 NPU Technical Reference Manual* or *Arm® Ethos™-U65 NPU Technical Reference Manual*.

8.6 CPU Private Region

Each processor in the system has its own copy of the CPU Private Region which is only accessible to itself. Each CPU Private Region consists of four subregions as follows:

`0x4001_0000` to `0x4001_FFFF`
: Implements a Non-secure Low Access Latency Region

`0x4801_0000` to `0x4801_FFFF`
: Implements a Non-secure High Access Latency Region

`0x5001_0000` to `0x5001_FFFF`
: Implements a Secure Low Access Latency Region

`0x5801_0000` to `0x5801_FFFF`
: Implements a Secure High Access Latency Region

Each of these regions is not accessible from any other manager in the system, including from the expansion subordinate interfaces on the Main and Peripheral Interconnect, except through the external debugger through the local CPUs. Of the four preceding regions only `0x4001_0000` to `0x4001_FFFF` and `0x5001_0000` to `0x5001_FFFF` implements any registers.

The memory map of the CPU Private Region is detailed in the following table.

The CPU Private Region address map defines the following values for security:

S
: Secure access only

NS
: Non-secure access only

P
: Privileged access only

UP
: Unprivileged and privileged access allowed

Table 8-43: CPU Private Region address map

Row ID	From address	To address	Size	Region name	Alias with row ID	Security	Description
0	`0x4001_0000`	`0x4001_1FFF`	-	-	-	-	-
1	`0x4001_2000`	`0x4001_2FFF`	4KB	CPU<n>_PWRCTRL	7	NS, P	CPU <n> Power Control block. See CPU<n>_PWRCTRL register block.
2	`0x4001_3000`	`0x4001_EFFF`		-			-
3	`0x4001_F000`	`0x4001_FFFF`	4KB	CPU<n>_IDENTITY	9	NS, UP	CPU <n> Identity block. See CPU<n>_IDENTITY register block.
4	`0x4801_0000`	`0x4801_FFFF`	-	-	-	-	-
5	`0x5001_0000`	`0x5001_0FFF`	-	-	-	-	-
6	`0x5001_1000`	`0x5001_1FFF`	4KB	CPU<n>_SECCTRL	-	S, P	CPU <n> Local Security Control block. See CPU<n>_SECCTRL register block.
7	`0x5001_2000`	`0x5001_2FFF`	4KB	CPU<n>_PWRCTRL	1	S, P	CPU <n> Power Control block. See CPU<n>_PWRCTRL register block.
8	`0x5001_3000`	`0x5001_EFFF`	-	-	-	-	-
9	`0x5001_F000`	`0x5001_FFFF`	4KB	CPU<n>_IDENTITY	3	S, UP	CPU <n> Identity block. See CPU<n>_IDENTITY register block.
10	`0x5801_0000`	`0x5801_FFFF`	-	-	-	-	-

8.6.1 CPU<n>_PWRCTRL register block

The CRSAS Ma2 implements a CPU<n>_PWRCTRL register block for each CPU<n> in the subsystem. All blocks reside at address `0x4001_2000` in a Non-secure region and are also aliased to `0x5001_2000` in the Secure region. Each CPU <n> can only see its own CPU<n>_PWRCTRL registers. These are read-only registers when accessed from the Non-secure region starting at address `0x4001_2000` and any writes access to it in that region are ignored.

The following table lists the registers in each CPU<n>_PWRCTRL register block.

Table 8-44: CPU<n>_PWRCTRL register map

Offset	Name	Type	Reset	Description
`0x000`	CPUPWRCFG	RW	`0x0000_0000`	CPU <n> Local Power Configuration register. See CPUPWRCFG.
`0x004 - 0xFCC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFD0`	PIDR4	RO	`0x0000_0004`	Peripheral ID 4
`0xFD4 - 0xFDC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFE0`	PIDR0	RO	`0x0000_005A`	Peripheral ID 0
`0xFE4`	PIDR1	RO	`0x0000_00B8`	Peripheral ID 1

Offset	Name	Type	Reset	Description
`0xFE8`	PIDR2	RO	`0x0000_000B`	Peripheral ID 2
`0xFEC`	PIDR3	RO	`0x0000_0000`	Peripheral ID 3
`0xFF0`	CIDR0	RO	`0x0000_000D`	Component ID 0
`0xFF4`	CIDR1	RO	`0x0000_00F0`	Component ID 1
`0xFF8`	CIDR2	RO	`0x0000_0005`	Component ID 2
`0xFFC`	CIDR3	RO	`0x0000_00B1`	Component ID 3

8.6.1.1 CPUPWRCFG

The CPUPWRCFG register provides the local CPU software control registers for power control.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Type

This register is read only if accessed from the Non-secure region starting at address `0x4001_2000` and any write accesses to it in that region are ignored.

Power domain

This register resides in the same power domain, PD_CPU<n>, as its associated CPU core so that when the CPU is powered down, the register is also powered down and is cleared when powered back up.

Reset domain

This register resides in the same reset domain, nWARMRESETCPU<n>, as its associated CPU core so that when the CPU is powered down, the register is also powered down and is cleared when powered back up.

Usage constraints

Each CPU <n> can only see its own CPU<n>_PWRCTRL registers.

Bit descriptions

Table 8-45: CPUPWRCFG bit descriptions

Bits	Name	Description	Type	Reset
[31:5]	-	Reserved	RAZ/ WI	`0x000_0000`

Bits	Name	Description	Type	Reset
[4]	TCM_MIN_PWR_STATE	Defines the minimum power state of the TCM for CPU<n>. • '0': OFF, • '1': Retention. This bit is read access only from the Non-secure world. When PD_CPU<n> returns from MEM_RET or MEM_RET_NOCACHE state to one of the ON states, this bit is set to 1'b1.	RW	`0x0`
[3:1]	-	Reserved	RAZ/WI	`0x0`
[0]	USEIWIC	When HIGH selects the use of IWIC for CPU <n> when in DeepSleep. Else selects the use of EWIC. If HASCPU<n>IWIC for this CPU <n> is 0, this field is reserved and **RAZ/WI**. This bit is read access only from the Non-secure world.	RW	`0x0`

8.6.2 CPU<n>_IDENTITY register block

The CRSAS Ma2 implements a CPU<n>_IDENTITY register block for each CPU <n> in the subsystem. All blocks reside at address `0x4001_F000` in a Non-secure region and are also aliased to `0x5001_F000` in the Secure region. Each CPU <n> can only see its own CPU<n>_IDENTITY registers. These are read-only registers and any write accesses to the region are ignored.

The following table lists the registers in each CPU<n>_IDENTITY block.

Table 8-46: CPU<n>_IDENTITY register map

Offset	Name	Type	Reset	Description
`0x000`	CPUID	RO	CPU<n>CPUIDRST	Unique CPU Identity Number, where <n> is used for the view that CPU <n> sees. See CPUID.
`0x004` – `0xFCC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFD0`	PIDR4	RO	`0x0000_0004`	Peripheral ID 4
`0xFD4` – `0xFDC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFE0`	PIDR0	RO	`0x0000_0055`	Peripheral ID 0
`0xFE4`	PIDR1	RO	`0x0000_00B8`	Peripheral ID 1
`0xFE8`	PIDR2	RO	`0x0000_000B`	Peripheral ID 2
`0xFEC`	PIDR3	RO	`0x0000_0000`	Peripheral ID 3
`0xFF0`	CIDR0	RO	`0x0000_000D`	Component ID 0
`0xFF4`	CIDR1	RO	`0x0000_00F0`	Component ID 1
`0xFF8`	CIDR2	RO	`0x0000_0005`	Component ID 2
`0xFFC`	CIDR3	RO	`0x0000_00B1`	Component ID 3

8.6.2.1 CPUID

The CPUID Register is a read only register that, when read by CPU <n>, provides an identity code to the CPU that is unique to that CPU.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Type

This register is read only and any write accesses to it are ignored.

Power domain

This register resides in the same power domain, PD_CPU<n>, as its associated CPU core so that when the CPU is powered down, the register is also powered down and is cleared when powered back up.

Reset domain

This register resides in the same reset domain, nWARMRESETCPU<n>, as its associated CPU core so that when the CPU is powered down, the register is also powered down and is cleared when powered back up.

Usage constraints

Each CPU <n> can only see its own CPU<n>_IDENTITY registers.

Bit descriptions

Table 8-47: CPUID bit descriptions

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RAZ/WI	`0x000_0000`
[3:0]	CPUID	CPU Identity. Defined by configuration CPU<n>CPUIDRST. The identity value for each CPU <n> must be unique.	RO	CPU<n>CPUIDRST

8.6.3 CPU<n>_SECCTRL register block

Each CPU <n> in the system has associated with it a CPU<n>_SECCTRL register block that allows the security locks of each CPU to be configured. Each register block resides in the same reset domain, nWARMRESETCPU<n>, and power domain as its associated CPU core so that when a CPU is powered down, they are also powered down and are cleared when powered back up. These registers are Secure access only and reside at address `0x5001_1000`.

The following table lists the registers in each CPU<n>_SECCTRL Register block.

Table 8-48: CPU<n>_SECCTRL Register Map

Offset	Name	Type	Reset	Description
`0x000`	CPUSECCFG	RW	`0x0000_0000`	CPU Local Security Configuration. See CPUSECCFG.
`0x004 - 0xFCC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFD0`	PIDR4	RO	`0x0000_0004`	Peripheral ID 4
`0xFD4 - 0xFDC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFE0`	PIDR0	RO	`0x0000_0059`	Peripheral ID 0
`0xFE4`	PIDR1	RO	`0x0000_00B8`	Peripheral ID 1
`0xFE8`	PIDR2	RO	`0x0000_001B`	Peripheral ID 2
`0xFEC`	PIDR3	RO	`0x0000_0000`	Peripheral ID 3
`0xFF0`	CIDR0	RO	`0x0000_000D`	Component ID 0
`0xFF4`	CIDR1	RO	`0x0000_00F0`	Component ID 1
`0xFF8`	CIDR2	RO	`0x0000_0005`	Component ID 2
`0xFFC`	CIDR3	RO	`0x0000_00B1`	Component ID 3

8.6.3.1 CPUSECCFG

The CPU Local Security Configuration Register allows software to set security lock bits at the interface of each associated CPU<n>.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Type

This register is Secure access only.

Power domain

This register resides in the same power domain as its associated CPU core so that when the CPU is powered down, the register is also powered down and is cleared when powered back up.

Reset domain

This register resides in the same reset domain, nWARMRESETCPU<n>, as its associated CPU core so that when the CPU is powered down, the register is also powered down and is cleared when powered back up.

Bit descriptions

Table 8-49: CPUSECCFG bit descriptions

Bits	Name	Description	Type	Reset
[31:6]	-	Reserved	RAZ/WI	`0x0000_0000`

Bits	Name	Description	Type	Reset
[5]	LOCKDTGU	When HIGH, disables writes to the CPU <n> DTGU_CTRL and DTGU_LUTn registers from software or from a debug agent connected to the processor. When set to HIGH, it cannot be cleared until reset.	RW1S	`0x0`
[4]	LOCKITGU	When HIGH, disables writes to the CPU <n> ITGU_CTRL and ITGU_LUTn from software or from a debug agent connected to the processor. When set to HIGH, it cannot be cleared until reset.	RW1S	`0x0`
[3]	LOCKTCM	When HIGH, disables writes to the CPU <n> ITCMCR, DTCMCR from software or from a debug agent connected to the processor. When set to HIGH, it cannot be cleared until reset.	RW1S	`0x0`
[2]	LOCKSMPU	When HIGH, disables write to the CPU <n> MPU_CTRL, MPU_RNR, MPU_RBAR, MPU_RLAR, MPU_RBAR_An, MPU_RLAR_An registers associated with the Secure MPU from software or from a debug agent connected to the processor. When set to HIGH, it cannot be cleared until reset.	RW1S	`0x0`
[1]	LOCKSAU	When HIGH, disables writes to the CPU <n> SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers from software or from a debug agent connected to the processor. When set to HIGH, it cannot be cleared until reset.	RW1S	`0x0`
[0]	LOCKSVTAIRCR	When HIGH, disables writes to the CPU <n> VTOR_S, AIRCR.PRIS, and AIRCR.BFHFNMINS registers. When set to HIGH, it cannot be cleared until reset.	RW1S	`0x0`

8.7 System Control Peripheral Region

The System Control Peripheral Regions are a collection of memory regions where system control related peripherals are mapped. These peripherals reside either in the PD_AON domain or in the PD_MGMT domain if PILEVEL = 2. There are four regions in total as follows:

`0x4002_0000` to `0x4003_FFFF`

Non-secure region for low latency system control peripherals. Some peripherals may be expected to be aliased in its associated Secure region, `0x5002_0000` to `0x5003_FFFF`.

`0x4802_0000` to `0x4803_FFFF`

Non-secure region for high latency system control peripherals. Some peripherals may be expected to be aliased in its associated Secure region, `0x5802_0000` to `0x5803_FFFF`.

`0x5002_0000` to `0x5003_FFFF`

Secure region for low latency system control peripherals. Some peripherals may be expected to be aliased in its associated Non-secure region, `0x4002_0000` to `0x4003_FFFF`.

`0x5802_0000` to `0x5803_FFFF`

Secure region for high latency system control peripherals. Some peripherals may be expected to be aliased in its associated Non-secure region, `0x4802_0000` to `0x4803_FFFF`.

For an aliased peripheral in these regions, mapping of each peripheral to either Secure or Non-secure region is determined by PPC that are controlled using the Secure Access Configuration register block. These PPCs also define privileged or unprivileged accessibility. For more information, Secure Access Configuration Register block.

The following table shows the System Control Peripheral Region address map.

The System Control Peripheral Region address map defines the following values for security:

NS_PPC
: Non-secure access only, gated by a PPC

S_PPC
: Secure access only, gated by a PPC

S
: Secure access only

NS
: Non-secure access only

P
: Privileged access only

UP
: Unprivileged and privileged access allowed

P_PPC
: Unprivileged access controlled by PPC

Note

When PPC protects a peripheral the configurability of the security or privileged attributes for a given peripheral is defined by PERIPHSPPPC0, PERIPHSPPPC1, PERIPHNSPPPC0, PERIPHNSPPPC1, PERIPHNSPPC0, and PERIPHNSPPC1 registers.

Table 8-50: System Control Peripheral Region address map

Row ID	From address	To address	Size	Region name	Alias with row ID	Security	Description
1	`0x4002_0000`	`0x4003_FFFF`	128KB	Reserved	-	-	Reserved
2	`0x4802_0000`	`0x4802_0FFF`	4KB	SYSINFO	7	NS, UP	System Information Register block. See SYSINFO register block.
3	`0x4802_1000`	`0x4802_EFFF`	56KB	Reserved	-	NS	Reserved When accessed, results in **RAZ/WI**.
4	`0x4802_F000`	`0x4802_FFFF`	4KB	SLOWCLK Timer	31	NS_PPC, P_PPC	Timer running on SLOWCLK. See SLOWCLK AON timers.
5	`0x4803_0000`	`0x4803_FFFF`	64KB	Reserved	-	-	Reserved
6	`0x5002_0000`	`0x5003_FFFF`	128KB	Reserved	-	-	Reserved
7	`0x5802_0000`	`0x5802_0FFF`	4KB	SYSINFO	2	S, UP	System Information Register block. See SYSINFO register block.
8	`0x5802_1000`	`0x5802_1FFF`	4KB	SYSCONTROL	-	S_PPC, P_PPC	System Control Register block. See System control register block.
18	`0x5802_2000`	`0x5802_2FFF`	4KB	SYS_PPU	-	S_PPC, P_PPC	PPU for BR_SYS. See Power Policy Units.
19	`0x5802_3000`	`0x5802_3FFF`	4KB	CPU0_PPU	-	S_PPC, P_PPC	PPU for BR_CPU0. See Power Policy Units.

Row ID	From address	To address	Size	Region name	Alias with row ID	Security	Description
20	`0x5802_4000`	`0x5802_4FFF`	4KB	CPU1_PPU	-	S_PPC, P_PPC	PPU for BR_CPU1. See Power Policy Units. When NUMCPU < 1, the CPU1_PPU region is reserved and RAZ/WI
21	`0x5802_5000`	`0x5802_5FFF`	4KB	CPU2_PPU	-	S_PPC, P_PPC	PPU for BR_CPU2. See Power Policy Units. When NUMCPU < 2, the CPU2_PPU region is reserved and **RAZ/WI**.
22	`0x5802_6000`	`0x5802_6FFF`	4KB	CPU3_PPU	-	S_PPC, P_PPC	PPU for BR_CPU3. See Power Policy Units. When NUMCPU < 3, the CPU3_PPU region is reserved and **RAZ/WI**.
23	`0x5802_7000`	`0x5802_7FFF`	4KB	Reserved	-	-	Reserved
24	`0x5802_8000`	`0x5802_8FFF`	4KB	MGMT_PPU	-	S_PPC, P_PPC	PPU for BR_MGMT. See Power Policy Units.
25	`0x5802_9000`	`0x5802_9FFF`	4KB	DEBUG_PPU	-	S_PPC, P_PPC	PPU for BR_DEBUG. See Power Policy Units.
26	`0x5802_A000`	`0x5802_AFFF`	4KB	NPU0_PPU	-	S_PPC, P_PPC	PPU for BR_NPU0. See Power Policy Units. When NUMNPU < 1, the NPU0_PPU region is reserved and **RAZ/WI**.
27	`0x5802_B000`	`0x5802_BFFF`	4KB	NPU1_PPU	-	S_PPC, P_PPC	PPU for BR_NPU1. See Power Policy Units. When NUMNPU < 2, the NPU1_PPU region is reserved and **RAZ/WI**.
28	`0x5802_C000`	`0x5802_CFFF`	4KB	NPU2_PPU	-	S_PPC, P_PPC	PPU for BR_NPU2. See Power Policy Units. When NUMNPU < 3, the NPU2_PPU region is reserved and **RAZ/WI**.
29	`0x5802_D000`	`0x5802_DFFF`	4KB	NPU3_PPU	-	S_PPC, P_PPC	PPU for BR_NPU3. See Power Policy Units. When NUMNPU < 4, the NPU3_PPU region is reserved and RAZ/WI
30	`0x5802_E000`	`0x5802_EFFF`	4KB	SLOWCLK Watchdog	-	S_PPC, P_PPC	Watchdog Timer running on SLOWCLK. See SLOWCLK AON timers.
31	`0x5802_F000`	`0x5802_FFFF`	4KB	SLOWCLK Timer	4	S_PPC, P_PPC	Timer running on SLOWCLK. See SLOWCLK AON timers.
32	`0x5803_0000`	`0x5803_FFFF`	64KB	Reserved	-	-	Reserved

8.7.1 SYSINFO register block

The System Information register block provides information on the system configuration and identity. This register block is read-only and is accessible by accesses of any security attributes. This module resides at base address `0x5802_0000` in the Secure region, and `0x4802_0000` in the Non-secure region.

Details of each register are described in the following table:

Table 8-51: System Information register map

Offset	Name	Type	Reset	Description
`0x000`	SOC_IDENTITY	RO	CFG_DEF	See SOC_IDENTITY.
`0x004`	SYS_CONFIG0	RO	CFG_DEF	System Hardware Configuration 0 Register. See SYS_CONFIG0.
`0x008`	SYS_CONFIG1	RO	CFG_DEF	System Hardware Configuration 1 Register. See SYS_CONFIG1.
`0x00C`	SYS_CONFIG2	RO	CFG_DEF	System Hardware Configuration 2 Register. See section SYS_CONFIG2.
`0x010 - 0xFC4`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFC8`	IIDR	RO	CFG_DEF	Subsystem Implementation Identity Register. See IIDR.
`0xFCC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFD0`	PIDR4	RO	`0x0000_0004`	Peripheral ID 4
`0xFD4 - 0xFDC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFE0`	PIDR0	RO	`0x0000_0058`	Peripheral ID 0
`0xFE4`	PIDR1	RO	`0x0000_00B8`	Peripheral ID 1
`0xFE8`	PIDR2	RO	`0x0000_002B`	Peripheral ID 2
`0xFEC`	PIDR3	RO	`0x0000_0000`	Peripheral ID 3
`0xFF0`	CIDR0	RO	`0x0000_000D`	Component ID 0
`0xFF4`	CIDR1	RO	`0x0000_00F0`	Component ID 1
`0xFF8`	CIDR2	RO	`0x0000_0005`	Component ID 2
`0xFFC`	CIDR3	RO	`0x0000_00B1`	Component ID 3

8.7.1.1 SOC_IDENTITY

The System-On-Chip (SoC) Identity register provides an area where software can find out about the part number, implementer, and revision number of the SoC. These are defined by configuration options that are expected to be set by a SoC integrator to identify the SoC.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Type

This register is read-only and is accessible by accesses of any security attributes.

Bit descriptions

Table 8-52: SOC_IDENTITY bit descriptions

Bits	Name	Description	Type	Reset
[31:20]	SOC_PRODUCT_ID	IMPL_DEF value identifying the SoC.	RO	SOCPRTID
[19:16]	SOC_VARIANT	IMPL_DEF value variant or major revision of the SoC.	RO	SOCVAR
[15:12]	SOC_REVISION	IMPL_DEF value used to distinguish minor revisions of the SoC.	RO	SOCREV

Bits	Name	Description	Type	Reset
[11:0]	SOC_IMPLEMENTATOR	Contains the JEP106 code of the company that implemented the SoC: • [11:8] JEP106 continuation code of implementer. • [7] Always 0. • [6:0] JEP106 identity code of implementer.	RO	SOCIMPLID

We recommend that the SoC Identity values are also used to identify the SoC or the complete subsystem to the debugger in one or both of the following ways:

1. If the system is a standalone system, the SoC Identity values can be used to generate the TARGETID of the SoC Debug Port in the expansion system, as follows:
    - TARGETID[31:28] uses SOCVAR,
    - TARGETID[27:16] uses SOCPRTID,
    - TARGETID[15:12] tied to `0x00`
    - TARGETID[11:1] uses {SOCIMPLID[11:8], SOCIMPLID[6:0]}.
    - TARGETID[0] tied to `0b1`.

    For more information on TARGETID, see *Arm® Debug Interface Architecture Specification ADIv6.0*.

2. The SoC Identity values can also be used in to define the PIDR values of the first Debug ROM in the expansion system that the debugger sees for this system, as follows:
    - REVISION uses SOCVAR,
    - {PART_1, PART_0} uses SOCPRTID,
    - {DES_2, DES_1, DES_0} uses SOCIMPLID,
    - REVAND uses SOCREV,
    - CMOD set to `0x0`.

    For more information on PIDR registers of CoreSight Debug ROM, see *Arm® CoreSight™ Architecture Specification v3.0*.

8.7.1.2 SYS_CONFIG0

The System Hardware Configuration registers provide several registers that allow software to query the configuration of the CRSAS Ma2 based subsystem.

See also SYS_CONFIG1 and SYS_CONFIG2.

In the following table, the fields CPU<*n*>_TCM_BANK_NUM and CPU<*n*>_HAS_SYSTCM refer to TCMs that are implemented on the system interconnect close to each associated CPU, rather than the TCMs that are implemented within the CPU core. The CRSAS Ma2 currently does not support TCMs being implemented on the system interconnect and hence these fields are reserved and RAZ/WI.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Type

This register is read-only and is accessible by accesses of any security attributes.

Bit descriptions

Table 8-53: SYS_CONFIG0 bit descriptions

Bits	Name	Description	Type	Reset
[31:28]	CPU1_TCM_BANK_NUM	The VM Bank that is the TCM memory for CPU 1.	RO	`0x0`
[27]	CPU1_HAS_SYSTCM	CPU 1 has System TCM: • '0': No • '1': Yes This is not the CPU's local ITCM or DTCM, but instead these are the TCMs that are implemented at system level.	RO	`0x0`
[26:24]	CPU1_TYPE	CPU 1 Core Type: • '000': Does not exist • '011': Cortex-M55 Processor • '100': Cortex-M85 • Others: Reserved	RO	CPU1TYPE
[23:20]	CPU0_TCM_BANK_NUM	The VM Bank that is the TCM memory for CPU 0.	RO	`0x0`
[19]	CPU0_HAS_SYSTCM	CPU 0 has System TCM: • '0' = No • '1' = Yes. This is not the CPU's local ITCM or DTCM, but instead these are the TCMs that are implemented at system level.	RO	`0x0`
[18:16]	CPU0_TYPE	CPU 0 Core Type: • '000': Does not exist • '011': Cortex-M55 Processor • '100': Cortex-M85 • Others: Reserved	RO	CPU0TYPE
[15:13]	-	Reserved	RO	`0x0`
[12:11]	PI_LEVEL	Power Infrastructure Level: • '00': Basic Level • '01': Intermediate Level • '10': Advanced Level • Others: Reserved.	RO	PILEVEL

Bits	Name	Description	Type	Reset
[10]	HAS_CSS	Includes CoreSight SoC-600M based Debug infrastructure. • '0': No • '1': Yes	RO	HASCSS
[9]	LCM_KMU_SAM_PRESENT	Includes LCM, KMU, and SAM: • '0': No • '1': Yes	RO	LCM_KMU_SAM_PRESENT
[8:4]	VM_ADDR_WIDTH	Volatile Memory Bank Address Width, where the size of each bank is equal to $2^{VM\_ADDR\_WIDTH}$ bytes.	RO	VMADDRWIDTH
[3:0]	NUM_VM_BANK	Number of Volatile Memory Banks.	RO	NUMVMBANK

8.7.1.3 SYS_CONFIG1

The System Hardware Configuration registers provide several registers to allow software to find out about the configuration of the CRSAS Ma2 based subsystem.

See also SYS_CONFIG0 and SYS_CONFIG2.

In the following table, the fields CPU<*n*>_TCM_BANK_NUM and CPU<*n*>_HAS_SYSTCM refer to TCMs that are implemented on the system interconnect close to each associated CPU, rather than the TCMs that are implemented within the CPU core. The CRSAS Ma2 currently does not support TCMs being implemented on the system interconnect and hence these fields are reserved and RAZ/WI.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Type

This register is read-only and is accessible by accesses of any security attributes.

Bit descriptions

Table 8-54: SYS_CONFIG1 bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RAZ/WI	`0x0000`
[15:12]	CPU3_TCM_BANK_NUM	The VM Bank that is the TCM memory for CPU 3.	RO	`0x0`

Bits	Name	Description	Type	Reset
[11]	CPU3_HAS_SYSTCM	CPU 3 has System TCM: • '0': No • '1': Yes Note that this is not the CPU's local ITCM or DTCM, but instead these are the TCMs that are implemented at system level.	RO	`0x0`
[10:8]	CPU3_TYPE	CPU 3 Core Type: • '000': Does not exist • '011': Cortex-M55 Processor • '100': Cortex-M85 • Others: Reserved.	RO	CPU3TYPE
[7:4]	CPU2_TCM_BANK_NUM	The VM Bank that is the TCM memory for CPU 2.	RO	`0x0`
[3]	CPU2_HAS_SYSTCM	CPU 2 has System TCM. • '0': No • '1': Yes Note that this is not the CPU's local ITCM or DTCM, but instead these are the TCMs that are implemented at system level.	RO	`0x0`
[2:0]	CPU2_TYPE	CPU 2 Core Type: • '000': Does not exist • '011': Cortex-M55 Processor • '100': Cortex-M85 • Others: Reserved.	RO	CPU2TYPE

8.7.1.4 SYS_CONFIG2

The System Hardware Configuration registers provide several registers to allow software to find out about the configuration of the CRSAS Ma2 based subsystem.

See also SYS_CONFIG0 and SYS_CONFIG1.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Type

This register is read-only and is accessible by accesses of any security attributes.

Bit descriptions

Table 8-55: SYS_CONFIG2 bit descriptions

Bits	Name	Description	Type	Reset
[31:15]	-	Reserved	RAZ/WI	`0x0000`
[14:12]	DMA_TYPE	DMA Core Type: • '000': Does not exist • '001': DMA-350 • Others: Reserved	RO	DMATYPE
[11:9]	NPU3_TYPE	NPU 3 Core Type: • '000': Does not exist • '001': Ethos-U55 • '010': Ethos-U65 • Others: Reserved	RO	NPU3TYPE
[8:6]	NPU2_TYPE	NPU 2 Core Type: • '000': Does not exist • '001': Ethos-U55 • '010': Ethos-U65 • Others: Reserved	RO	NPU2TYPE
[5:3]	NPU1_TYPE	NPU 1 Core Type: • '000': Does not exist • '001': Ethos-U55 • '010': Ethos-U65 • Others: Reserved	RO	NPU1TYPE
[2:0]	NPU0_TYPE	NPU 0 Core Type: • '000': Does not exist • '001': Ethos-U55 • '010': Ethos-U65 • Others: Reserved	RO	NPU0TYPE

8.7.1.5 IIDR

The subsystem Implementation Identity register provides an area where software can find out about the subsystem Implementation part number, its implementer and revision number. These are defined by configuration options that are expected to be set by the subsystem implementer.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Type

This register is read-only and is accessible by accesses of any security attributes.

Bit descriptions

Table 8-56: IIDR bit descriptions

Bits	Name	Description	Type	Reset
[31:20]	IMP_PRODUCT_ID	IMPL_DEF value identifying the subsystem implementation.	RO	IMPLPRTID
[19:16]	IMP_VARIANT	IMPL_DEF value variant or major revision of the subsystem implementation.	RO	IMPLVAR
[15:12]	IMP_REVISION	IMPL_DEF value used to distinguish minor revisions of the subsystem implementation.	RO	IMPLREV
[11:0]	IMP_IMPLEMENTATOR	Contains the JEP106 code of the company that implemented the subsystem implementation: • [11:8] JEP106 continuation code of implementer. • [7] Always 0. • [6:0] JEP106 identity code of implementer.	RO	IMPLID

8.7.2 System Control register block

The System Control register block implements registers for power, clocks, resets, and other general system control. This module resides at base address `0x5802_1000` in the Secure region. The System Control register block is Secure privileged access only. For write access to these registers, only 32-bit writes are supported. Any byte and halfword writes results in its write data ignored.

Most System Control registers reside in the PD_MGMT power domain when PILEVEL = 2, or is merged into PD_AON when PILEVEL < 2. When entering lower power states, some of the register values must be retained. How retention is achieved is **IMPLEMENTATION DEFINED**. Other registers not in PD_MGMT reside instead in the PD_AON domain.

The following table shows the details of this register block:

Table 8-57: System Control register map

Offset	Name	Type	Reset	Description
`0x000`	SECDBGSTAT	RO	CFG_DEF	Secure Debug Configuration Status Register. See SECDBGSTAT.
`0x004`	SECDBGSET	RW	`0x0000_0000`	Secure Debug Configuration Set Register. See SECDBGSET.
`0x008`	SECDBGCLR	WO	`0x0000_0000`	Secure Debug Configuration Clear Register. See SECDBGCLR.
`0x00C`	SCSECCTRL	RW	`0x0000_0000`	System Control Security Controls Register. See SCSECCTRL.
`0x010`	CLK_CFG0	RW	CFG_DEF	Clock Configuration Register 0. See CLK_CFG0.

Offset	Name	Type	Reset	Description
0x014	CLK_CFG1	RW	CFG_DEF	Clock Configuration Register 1. See CLK_CFG1.
0x018	CLOCK_FORCE	RW	CFG_DEF	Clock Forces. See CLOCK_FORCE.
0x1C	CLK_CFG2	RW	CFG_DEF	Clock Configuration Register 1. See CLK_CFG2.
0x020 – 0x0FF	-	RAZ/WI	0x0000_0000	Reserved
0x100	RESET_SYNDROME	RW	0x0000_0001	Reset syndrome. See RESET_SYNDROME.
0x104	RESET_MASK	RW	CFG_DEF	Reset Mask. See RESET_MASK.
0x108	SWRESET	WO	0x0000_0000	Software Reset. See SWRESET.
0x10C	GRETREG	RW	0x0000_0000	General Purpose Retention Register. See GRETREG.
0x110	INITSVTOR0[1]	RW	CFG_DEF	CPU 0 Initial Secure Reset Vector Register. See INITSVTOR<n>.
0x114	INITSVTOR1[1]	RW	CFG_DEF	CPU 1 Initial Secure Reset Vector Register. See INITSVTOR<n>.
0x118	INITSVTOR2[2]	RW	CFG_DEF	CPU 2 Initial Secure Reset Vector Register. See INITSVTOR<n>.
0x11C	INITSVTOR3[3]	RW	CFG_DEF	CPU 3 Initial Secure Reset Vector Register. See INITSVTOR<n>.
0x120	CPUWAIT	RW	CFG_DEF	CPU Boot Wait Control. See CPUWAIT.
0x124	NMI_ENABLE	RW	CFG_DEF	Enabling and Disabling Non Maskable Interrupts. See NMI_ENABLE.
0x128	PPUINTSTAT	RO	0x0000_0000	PPU Interrupt Status. See PPUINTSTAT.
0x12C – 0x1F8	-	RAZ/WI	0x0000_0000	Reserved
0x1FC	PWRCTRL	RW	0x0000_0003	Power Configuration and Control. See PWRCTRL.
0x200	PDCM_PD_SYS_ SENSE	RW	CFG_DEF	PDCM PD_SYS Sensitivity. See PDCM_PD_SYS_SENSE.

Offset	Name	Type	Reset	Description
0x204	PDCM_PD_CPU0_ SENSE	RO	0x0000_0000	PDCM PD_CPU0 Sensitivity. See PDCM_PD_CPU<n>_SENSE.
0x208	PDCM_PD_CPU1_ SENSE[1]	RO	0x0000_0000	PDCM PD_CPU1 Sensitivity. See PDCM_PD_CPU<n>_SENSE.
0x20C	PDCM_PD_CPU2_ SENSE[2]	RO	0x0000_0000	PDCM PD_CPU2 Sensitivity. See PDCM_PD_CPU<n>_SENSE.
0x210	PDCM_PD_CPU3_ SENSE[3]	RO	0x0000_0000	PDCM PD_CPU3 Sensitivity. See PDCM_PD_CPU<n>_SENSE.
0x214	PDCM_PD_VMR0_ SENSE[4]	RW	0x4000_0000	PDCM PD_VMR0 Sensitivity. See PDCM_PD_VMR<x>_SENSE.
0x218	PDCM_PD_VMR1_ SENSE[5]	RW	0x4000_0000	PDCM PD_VMR1 Sensitivity. See PDCM_PD_VMR<x>_SENSE.
0x21C	PDCM_PD_VMR2_ SENSE[6]	RW	0x4000_0000	PDCM PD_VMR2 Sensitivity. See PDCM_PD_VMR<x>_SENSE.
0x220	PDCM_PD_VMR3_ SENSE[7]	RW	0x4000_0000	PDCM PD_VMR3 Sensitivity. See PDCM_PD_VMR<x>_SENSE.
0x224 - 0x248	-	RAZ/WI	0x0000_0000	Reserved
0x24C	PDCM_PD_MGMT_ SENSE	RW	CFG_DEF	PDCM PD_MGMT Sensitivity. See PDCM_PD_MGMT_SENSE.
0x250 - 0x258	-	RAZ/WI	0x0000_0000	Reserved
0x25C	LCM_DCU_FORCE _DISABLE	RW	CFG_DEF	LCM DCU Force Disable See LCM_DCU_FORCE_DISABLE
0x260 - 0xFCC	-	RAZ/WI	0x0000_0000	Reserved
0xFD0	PIDR4	RO	0x0000_0004	Peripheral ID 4
0xFD4 - 0xFDC	-	RAZ/WI	0x0000_0000	Reserved
0xFE0	PIDR0	RO	0x0000_0054	Peripheral ID 0
0xFE4	PIDR1	RO	0x0000_00B8	Peripheral ID 1
0xFE8	PIDR2	RO	0x0000_004B	Peripheral ID 2
0xFEC	PIDR3	RO	0x0000_0000	Peripheral ID 3
0xFF0	CIDR0	RO	0x0000_000D	Component ID 0
0xFF4	CIDR1	RO	0x0000_00F0	Component ID 1
0xFF8	CIDR2	RO	0x0000_0005	Component ID 2
0xFFC	CIDR3	RO	0x0000_00B1	Component ID 3

[1] These registers do not exist and are reserved if NUMCPU < 1:

- PDCM_PD_CPU1_SENSE
- INITSVTOR0

- INITSVTOR1

[2] These registers do not exist and are reserved if NUMCPU < 2:

- INITSVTOR2
- PDCM_PD_CPU2_SENSE

[3] These registers do not exist and are reserved if NUMCPU < 3:

- INITSVTOR3
- PDCM_PD_CPU3_SENSE

[4] These registers do not exist and are reserved if NUMVMBANK < 1:

- PDCM_PD_VMR0_SENSE

[5] These registers do not exist and are reserved if NUMVMBANK < 2:

- PDCM_PD_VMR1_SENSE

[6] These registers do not exist and are reserved if NUMVMBANK < 3:

- PDCM_PD_VMR2_SENSE

[7] These registers do not exist and are reserved if NUMVMBANK < 4:

- PDCM_PD_VMR3_SENSE

8.7.2.1 Secure Debug Configuration Registers

The Secure Debug Configuration Registers are used to select the source value for the Secure Debug Authentication, DBGEN, NIDEN, SPIDEN, SPNIDEN, DAPACCEN, and Debug Access Controls, DAPDSSACCEN, SYSDSSACCENX, and SYSDSSACCEN<n>. For each signal and just one for all SYSDSSACCEN<n> and SYSDSSACCENX, 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 the internal register and selector values by setting the associated bit in the SECDBGSET register or in the SECDBGCLR register respectively. Secure software can read the output values used system wide by reading the associated SECDBGSTAT register bit. Secure software can read internal register values by reading SECDBGSET, see SECDBGSET.

For example, the source of DBGEN value used in the system is selected by the DBGEN_SEL where:

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

Write 1 to the SECDBGSET.DBGEN_I_SET register to set DBGEN_I value to HIGH. Write 1 to SECDBGSET.DBGEN_SEL_SET to set DBGEN_SEL value to HIGH.

Write 1 to the SECDBGCLR.DBGEN_I_CLR register to set DBGEN_I value to LOW. Write 1 to SECDBGCLR.DBGEN_SEL_CLR to set DBGEN_SEL to LOW.

Read the SECDBGSTAT.DBGEN_STATUS register to read the output value of DBGEN. Read SECDBGSET.DBGEN_I_SET register to read the value of DBGEN_I.

The DBGEN value is also made available to external expansion logic through the DBGEN output signal of the subsystem.

Selector Disable Configuration options are provided to allow each of the selector to be forced to zero, forcing the associated SEL_STATUS field to LOW, forcing each respective debug control output to use its external value:

- DBGENSELDIS for disabling DBGEN_SEL,
- NIDENSELDIS for disabling NIDEN_SEL,
- SPIDENSELDIS for disabling SPIDEN_SEL,
- SPNIDENSELDIS for disabling SPNIDEN_SEL,
- DAPACCENSELDIS for disabling DAPACCEN_SEL,
- DAPDSSACCENSELDIS for disabling DAPDSSACCEN_SEL,
- SYSDSSACCENSELDIS for disabling SYSDSSACCEN_SEL.

These can be used to disable the ability for Secure firmware to modify or override the Debug Authentication and the Debug Access Controls values. When the Lifecycle Manager exists (LCM_KMU_SAM_PRESENT = ‘1’) in the system dedicated bits of LCM Debug Control Unit DCUEN output controls the debug authentication instead.

These registers are reset by nCOLDRESETAON only. These registers reside in the PD_AON power domain but can also reside in PD_MGMT power domain if its states are saved and restored when entering and then leaving the lower power state, respectively. This choice is **IMPLEMENTATION DEFINED**.

8.7.2.1.1 SECDBGSTAT

Secure Debug Configuration Status Register.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain, if its states are saved and restored when entering and then leaving the lower power state, respectively. This choice is **IMPLEMENTATION DEFINED**.

Reset domain

nCOLDRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-58: SECDBGSTAT bit descriptions

Bits	Name	Description	Type	Reset
[31]	-	Reserved	RAZ/WI	`0x0`
[30]	SYSDSSACCENSELDIS_STATUS	Returns the SYSDSSACCENSELDIS configuration value when read. Ignores Writes. Reserved and **RAZ/WI** if HASCSS = 0.	RO	SYSDSSACCENSELDIS
[29]	DAPDSSACCENSELDIS_STATUS	Returns the DAPDSSACCENSELDIS configuration value when read.	RO	DAPDSSACCENSELDIS
[28]	DAPACCENSELDIS_STATUS	Returns the DAPACCENSELDIS configuration value when read.	RO	DAPACCENSELDIS
[27]	SPNIDENSELDIS_STATUS	Returns the SPNIDENSELDIS configuration value when read.	RO	SPNIDENSELDIS
[26]	SPIDENSELDIS_STATUS	Returns the SPIDENSELDIS configuration value when read.	RO	SPIDENSELDIS
[25]	NIDENSELDIS_STATUS	Returns the NIDENSELDIS configuration value when read.	RO	NIDENSELDIS
[24]	DBGENSELDIS_STATUS	Returns the DBGENSELDIS configuration value when read.	RO	DBGENSELDIS
[23:18]	-	Reserved	RAZ/WI	`0x00`
[17]	SYSDSSACCEN_SEL_STATUS	Active-HIGH System Mapped Debug Access Enable Selector Value. Used as a common selector for all SYSDSSACCEN<*n*>_STATUS. This bit returns the SYSDSSACCEN_SEL value Forced to Zero if SYSDSSACCENSELDIS = 1. Reserved and **RAZ/WI** if HASCSS = 0.	RO	`0x0`
[16]	SYSDSSACCENX_STATUS	Active-HIGH System Mapped Debug Access for Implementation Defined Manager(s). Reserved and **RAZ/WI** if HASCSS = 0. This bit reflects the value on the SYSDSSACCENX pin.	RO	SYSDSSACCENX
[15]	SYSDSSACCEN3_STATUS	Active-HIGH System Mapped Debug Access for CPU 3 Enable Value. This bit reflects the value on the SYSDSSACCEN3 after selection. Reserved and **RAZ/WI** if HASCSS = 0 or NUMCPU < 3.	RO	SYSDSSACCEN3
[14]	SYSDSSACCEN2_STATUS	Active-HIGH System Mapped Debug Access for CPU 2 Enable Value. This bit reflects the value on the SYSDSSACCEN2 after selection. Reserved and **RAZ/WI** if HASCSS = 0 or NUMCPU < 2.	RO	SYSDSSACCEN2
[13]	SYSDSSACCEN1_STATUS	Active-HIGH System Mapped Debug Access for CPU 1 Enable Value. This bit reflects the value on the SYSDSSACCEN1 after selection. Reserved and **RAZ/WI** if HASCSS = 0 or NUMCPU < 1.	RO	SYSDSSACCEN1

Bits	Name	Description	Type	Reset
[12]	SYSDSSACCEN0_STATUS	Active-HIGH System Mapped Debug Access for CPU 0 Enable Value. This bit reflects the value on the SYSDSSACCEN0 after selection. Reserved and **RAZ/WI** if HASCSS = 0.	RO	SYSDSSACCEN0
[11]	DAPDSSACCEN_SEL_STATUS	Active-HIGH DAP to Debug Subsystem Access Enable Selector Value. This bit returns the DAPDSSACCEN_SEL value. Forced to Zero if DAPDSSACCENSELDIS = 1.	RO	`0x0`
[10]	DAPDSSACCEN_STATUS	Active-HIGH DAP to Debug Subsystem Access Enable Value. This bit reflects the value on the DAPDSSACCEN pin.	RO	DAPDSSACCEN
[9]	DAPACCEN_SEL_STATUS	Active-HIGH DAP Access Enable Selector Value. This bit returns the DAPACCEN_SEL value. Forced to Zero if DAPACCENSELDIS = 1.	RO	`0x0`
[8]	DAPACCEN_STATUS	Active-HIGH DAP Access Enable Value. This bit reflects the value on the DAPACCEN pin.	RO	DAPACCEN
[7]	SPNIDEN_SEL_STATUS	Active-HIGH Secure Privileged Non-Invasive Debug Enable Selector Value. This bit returns the SPNIDEN_SEL value. Forced to Zero if SPNIDENSELDIS = 1.	RO	`0x0`
[6]	SPNIDEN_STATUS	Active-HIGH Secure Privileged Non-Invasive Debug Enable Value. This bit reflects the value on the SPNIDEN pin.	RO	SPNIDEN
[5]	SPIDEN_SEL_STATUS	Active-HIGH Secure Privileged Invasive Debug Enable Selector Value. This bit returns the SPIDEN_SEL value. Forced to Zero if SPIDENSELDIS = 1.	RO	`0x0`
[4]	SPIDEN_STATUS	Active-HIGH Secure Privileged Invasive Debug Enable Value. This bit reflects the value on the SPIDEN pin.	RO	SPIDEN
[3]	NIDEN_SEL_STATUS	Active-HIGH Non-Invasive Debug Enable Selector Value. This bit returns the NIDEN_SEL value. Forced to Zero if NIDENSELDIS = 1.	RO	`0x0`
[2]	NIDEN_STATUS	Active-HIGH Non-Invasive Debug Enable Value. This bit reflects the value on the NIDEN pin.	RO	NIDEN
[1]	DBGEN_SEL_STATUS	Active-HIGH Debug Enable Selector Value. This bit returns the DBGEN_SEL value. Forced to Zero if DBGENSELDIS = 1.	RO	`0x0`
[0]	DBGEN_STATUS	Active-HIGH Debug Enable Value. This bit reflects the value on the DBGEN pin.	RO	DBGEN

8.7.2.1.2 SECDBGSET

Secure Debug Configuration Set register.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain if its states are saved and restored when entering and then leaving the lower power state, respectively. This choice is **IMPLEMENTATION DEFINED**.

Reset domain

nCOLDRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-59: SECDBGSET bit descriptions

Bits	Name	Description	Type	Reset
[31:18]	-	Reserved	RAZ/WI	`0x0000`
[17]	SYSDSSACCEN_SEL_SET	Set Active-HIGH System Mapped Debug Access Enable Selector Value. Write HIGH to set SYSDSSACCEN_SEL. Reserved and **RAZ/WI** if HASCSS = 0 or SYSDSSACCENSELDIS = 1.	RAZW1S	`0x0`
[16]	SYSDSSACCENX_I_SET	Set internal version of Active-HIGH System Mapped Debug Access for Implementation Defined Manager(s) Enable. Write HIGH to set SYSDSSACCENX_I. When read returns SYSDSSACCENX_I. Reserved and **RAZ/WI** if HASCSS = 0 or SYSDSSACCENSELDIS = 1.	RW1S	`0x0`
[15]	SYSDSSACCEN3_I _SET	Set internal version of Active-HIGH System Mapped Debug Access for CPU 3 Enable Set Register. Write HIGH to set SYSDSSACCEN3_I. When read returns SYSDSSACCEN3_I. Reserved and **RAZ/WI** if HASCSS = 0 or NUMCPU < 3 or SYSDSSACCENSELDIS = 1. When read, this returns the internal SYSDSSACCEN3 register value before selection.	RW1S	`0x0`
[14]	SYSDSSACCEN2_I _SET	Set internal version of Active-HIGH System Mapped Debug Access for CPU 2 Enable. Write HIGH to set SYSDSSACCEN2_I. When read returns SYSDSSACCEN2_I. Reserved and **RAZ/WI** if HASCSS = 0 or NUMCPU < 2 or SYSDSSACCENSELDIS = 1.	RW1S	`0x0`
[13]	SYSDSSACCEN1_I _SET	Set internal version of Active-HIGH System Mapped Debug Access for CPU 1 Enable. Write HIGH to set SYSDSSACCEN1_I. When read returns SYSDSSACCEN1_I. Reserved and **RAZ/WI** if HASCSS = 0 or NUMCPU < 1 or SYSDSSACCENSELDIS = 1.	RW1S	`0x0`
[12]	SYSDSSACCEN0_I _SET	Set internal version of Active-HIGH System Mapped Debug Access for CPU 0 Enable. Write HIGH to set SYSDSSACCEN0_I. When read returns SYSDSSACCEN0_I. Reserved and **RAZ/WI** if HASCSS = 0 or SYSDSSACCENSELDIS = 1.	RW1S	`0x0`
[11]	DAPDSSACCEN_SEL_SET	Set Active-HIGH DAP to Debug Subsystem Access Enable Selector. Write HIGH to set DAPDSSACCEN_SEL. **RAZ/WI** if DAPDSSACCENSELDIS = 1.	RAZW1S	`0x0`

Bits	Name	Description	Type	Reset
[10]	DAPDSSACCEN_I _SET	Set internal version of Active-HIGH DAP to Debug Subsystem Access Enable. Write HIGH to set DAPDSSACCEN_I. When read returns DAPDSSACCEN_I. **RAZ/WI** if DAPDSSACCENSELDIS = 1.	RW1S	0x0
[9]	DAPACCEN_SEL_SET	Set Active-HIGH DAP Access Enable Selector. Write HIGH to set DAPACCEN_SEL. **RAZ/WI** if DAPACCENSELDIS = 1.	RAZW1S	0x0
[8]	DAPACCEN_I _SET	Set internal version of Active-HIGH DAP Access Enable. Write HIGH to set DAPACCEN_I. When read returns DAPACCEN_I. **RAZ/WI** if DAPACCENSELDIS = 1.	RW1S	0x0
[7]	SPNIDEN_SEL_SET	Set Active-HIGH Secure Privileged Non-Invasive Debug Enable Selector. Write HIGH to set SPNIDEN_SEL. **RAZ/WI** if SPNIDENSELDIS = 1.	RAZW1S	0x0
[6]	SPNIDEN_I _SET	Set internal version of Active-HIGH Secure Privileged Non-Invasive Debug Enable. Write HIGH to set SPNIDEN_I. When read returns SPNIDEN_I. **RAZ/WI** if SPNIDENSELDIS = 1.	RW1S	0x0
[5]	SPIDEN_SEL_SET	Set Active-HIGH Secure Privileged Invasive Debug Enable Selector. Write HIGH to set SPIDEN_SEL. **RAZ/WI** if SPIDENSELDIS = 1.	RAZW1S	0x0
[4]	SPIDEN_I _SET	Set internal version of Active-HIGH Secure Privileged Invasive Debug Enable. Write HIGH to set SPIDEN_I. When read returns SPIDEN_I. **RAZ/WI** ifSPIDENSELDIS = 1.	RW1S	0x0
[3]	NIDEN_SEL_SET	Set Active-HIGH Non-Invasive Debug Enable Selector. Write HIGH to set NIDEN_SEL. **RAZ/WI** if NIDENSELDIS = 1.	RAZW1S	0x0
[2]	NIDEN_I _SET	Set internal version of Active-HIGH Non-Invasive Debug Enable. Write HIGH to set NIDEN_I. When read returns NIDEN_I. **RAZ/WI** if NIDENSELDIS = 1.	RW1S	0x0
[1]	DBGEN_SEL_SET	Set Active-HIGH Debug Enable Selector. Write HIGH to set DBGEN_SEL. **RAZ/WI** if DBGENSELDIS = 1.	RAZW1S	0x0
[0]	DBGEN_I_SET	Set internal version of Active-HIGH Debug Enable. Write HIGH to set DBGEN_I. When read returns DBGEN_I. **RAZ/WI** if DBGENSELDIS = 1.	RW1S	0x0

8.7.2.1.3 SECDBGCLR

The Secure Debug Configuration Clear register.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain if its states are saved and restored when entering and then leaving the lower power state, respectively. This choice is **IMPLEMENTATION DEFINED**.

Reset domain

nCOLDRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-60: SECDBGCLR bit descriptions

Bits	Name	Description	Type	Reset
[31:18]	-	Reserved	RAZ/WI	0x0000
[17]	SYSDSSACCEN_SEL_CLR	Clears Active-HIGH System Mapped Debug Access Enable Selector Value. Write HIGH to clear SYSDSSACCEN_SEL. **RAZ/WI** if SYSDSSACCENSELDIS = 1. Reserved and **RAZ/WI** if HASCSS = 0.	RAZ/W1C	0x0
[16]	SYSDSSACCENX_I _CLR	Clears internal version of Active High System Mapped Debug Access for Implementation Defined Manager(s) Enable. Write HIGH to clear SYSDSSACCENX_I. **RAZ/WI** if SYSDSSACCENSELDIS = 1. Reserved and **RAZ/WI** if HASCSS = 0.	RAZ/W1C	0x0
[15]	SYSDSSACCEN3_I _CLR	Clears internal version of Active High System Mapped Debug Access for CPU 3 Enable. Write HIGH to clear SYSDSSACCEN3_I. **RAZ/WI** if SYSDSSACCENSELDIS = 1. Reserved and **RAZ/WI** if HASCSS = 0 or NUMCPU < 3.	RAZ/W1C	0x0
[14]	SYSDSSACCEN2_I _CLR	Clears internal version of Active High System Mapped Debug Access for CPU 2 Enable. Write HIGH to clear SYSDSSACCEN2_I. **RAZ/WI** if SYSDSSACCENSELDIS = 1. Reserved and **RAZ/WI** if HASCSS = 0 or NUMCPU < 2.	RAZ/W1C	0x0
[13]	SYSDSSACCEN1_I _CLR	Clears internal version of Active High System Mapped Debug Access for CPU 1 Enable. Write HIGH to clear SYSDSSACCEN1_I. **RAZ/WI** if SYSDSSACCENSELDIS = 1. Reserved and **RAZ/WI** if HASCSS = 0 or NUMCPU < 1.	RAZ/W1C	0x0
[12]	SYSDSSACCEN0_I _CLR	Clears internal version of Active High System Mapped Debug Access for CPU 0 Enable. Write HIGH to clear SYSDSSACCEN0_I. **RAZ/WI** if SYSDSSACCENSELDIS = 1. Reserved and **RAZ/WI** if HASCSS = 0.	RAZ/W1C	0x0
[11]	DAPDSSACCEN_SEL_CLR	Clears Active-HIGH DAP to Debug Subsystem Access Enable Selector. Write HIGH to clear DAPDSSACCEN_SEL. **RAZ/WI** if DAPDSSACCENSELDIS = 1.	RAZ/W1C	0x0
[10]	DAPDSSACCEN_I _CLR	Clears internal version of Active High DAP to Debug Subsystem Access Enable. Write HIGH to clear DAPDSSACCEN_I. **RAZ/WI** if DAPDSSACCENSELDIS = 1.	RAZ/W1C	0x0
[9]	DAPACCEN_SEL_CLR	Clears Active-HIGH DAP Access Enable Selector. Write HIGH to clear DAPACCEN_SEL. **RAZ/WI** if DAPACCENSELDIS = 1.	RAZ/W1C	0x0
[8]	DAPACCEN_I _CLR	Clears internal version of Active High DAP Access Enable. Write HIGH to clear DAPACCEN_I. **RAZ/WI** if DAPACCENSELDIS = 1.	RAZ/W1C	0x0
[7]	SPNIDEN_SEL_CLR	Clears Active-HIGH Secure Privileged Non-Invasive Debug Enable Selector. Write HIGH to clear SPNIDEN_SEL. **RAZ/WI** if SPNIDENSELDIS = 1.	RAZ/W1C	0x0
[6]	SPNIDEN_I _CLR	Clears internal version of Active High Secure Privileged Non-Invasive Debug Enable. Write HIGH to clear SPNIDEN_I. **RAZ/WI** if SPNIDENSELDIS = 1.	RAZ/W1C	0x0
[5]	SPIDEN_SEL_CLR	Clears Active-HIGH Secure Privileged Invasive Debug Enable Selector. Write HIGH to clear SPIDEN_SEL. **RAZ/WI** if SPIDENSELDIS = 1.	RAZ/W1C	0x0
[4]	SPIDEN_I _CLR	Clears internal version of Active High Secure Privileged Invasive Debug Enable. Write HIGH to clear SPIDEN_I. **RAZ/WI** if SPIDENSELDIS = 1.	RAZ/W1C	0x0
[3]	NIDEN_SEL_CLR	Clears Active-HIGH Non-Invasive Debug Enable Selector. Write HIGH to clear NIDEN_SEL. **RAZ/WI** if NIDENSELDIS = 1.	RAZ/W1C	0x0
[2]	NIDEN_I _CLR	Clears internal version of Active High Non-Invasive Debug Enable. Write HIGH to clear NIDEN_I. **RAZ/WI** if NIDENSELDIS = 1.	RAZ/W1C	0x0

Bits	Name	Description	Type	Reset
[1]	DBGEN_SEL_CLR	Clears Active-HIGH Debug Enable Selector. Write HIGH to clear DBGEN_SEL. **RAZ/WI** if DBGENSELDIS = 1.	RAZ/ W1C	0x0
[0]	DBGEN_I _CLR	Clears internal version of Active High Debug Enable. Write HIGH to clear DBGEN_I. **RAZ/WI** if DBGENSELDIS = 1.	RAZ/ W1C	0x0

8.7.2.2 SCSECCTRL

The System Control Security Controls provides register bits to set the Secure Configuration lock of this register block.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

These registers reside in the PD_AON power domain but can also reside in PD_MGMT power domain if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nCOLDRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-61: SCSECCTRL bit descriptions

Bits	Name	Description	Type	Reset
[31:3]	-	Reserved	RAZ/ WI	0x0000_0000
[2]	SCSECCFGLOCK	Active-HIGH control to disable writes to Security related control registers SECDBGSET and SECDBGCLR. When set to HIGH, it can no longer be cleared to zero except through Cold reset.	RW1S	0x0
[1:0]	-	Reserved	RAZ/ WI	0x0

8.7.2.3 CLK_CFG0

The CLK_CFG0 register provides control register fields to drive expansion clock generation logic that drives clock for this subsystem.

Each clock control handshake comprises of a configuration request register, that is CLKCFG, and a status register, that is CLKCFGSTATUS that acts as an acknowledgment. After writing to each CLKCFG field, the software has to poll the associated CLKCFGSTATUS field until the CLKCFGSTATUS is the same as the CLKCFG before doing any other operations. In addition, if it is the first write to the register after boot, we recommend that software polls to check that the targeted CLKCFGSTATUS field matches its associated CLKCFG value before the write occurs. The actual number of bits being implemented in the related CLKCFG and CLKCFGSTATUS signals can be less than that defined below, and is IMPLEMENTATION DEFINED. For any bit that is not implemented, the associated register bit is RAZ/WI.

See also CLK_CFG1 and CLK_CFG2.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nCOLDRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-62: CLK_CFG0 bit descriptions

Bits	Name	Description	Type	Reset
[31:28]	CPU3CLKCFGSTATUS	Clock Configuration Status value that reports the status of clock control for CPU3CLK. **RAZ/WI** if NUMCPU < 3	RO	CPU3CLKCFGSTATUS
[27:24]	CPU2CLKCFGSTATUS	Clock Configuration Status value that reports the status of clock control for CPU2CLK. **RAZ/WI** if NUMCPU < 2	RO	CPU2CLKCFGSTATUS
[23:20]	CPU1CLKCFGSTATUS	Clock Configuration Status value that reports the status of clock control for CPU1CLK. **RAZ/WI** if NUMCPU < 1	RO	CPU1CLKCFGSTATUS
[19:16]	CPU0CLKCFGSTATUS	Clock Configuration Status value that reports the status of clock control for CPU0CLK.	RO	CPU0CLKCFGSTATUS

Bits	Name	Description	Type	Reset
[15:12]	CPU3CLKCFG	Clock Configuration value that drives CPU3CLKCFG signals. **RAZ/WI** if NUMCPU < 3	RW	CPU3CLKCFGRST
[11:8]	CPU2CLKCFG	Clock Configuration value that drives CPU2CLKCFG signals. **RAZ/WI** if NUMCPU < 2	RW	CPU2CLKCFGRST
[7:4]	CPU1CLKCFG	Clock Configuration value that drives CPU1CLKCFG signals. **RAZ/WI** if NUMCPU < 1	RW	CPU1CLKCFGRST
[3:0]	CPU0CLKCFG	Clock Configuration value that drives CPU0CLKCFG signals.	RW	CPU0CLKCFGRST

8.7.2.4 CLK_CFG1

The CLK_CFG1 register provides control register fields to drive expansion clock generation logic that drives clock for this subsystem.

Each clock control handshake comprises of a configuration request register, that is CLKCFG, and a status register, that is CLKCFGSTATUS that acts as an acknowledgment. After writing to each CLKCFG field, the software has to poll the associated CLKCFGSTATUS field until the CLKCFGSTATUS is the same as the CLKCFG before doing any other operations. In addition, if it is the first write to the register after boot, we recommend that software polls to check that the targeted CLKCFGSTATUS field matches its associated CLKCFG value before the write occurs. The actual number of bits being implemented in the related CLKCFG and CLKCFGSTATUS signals can be less than that defined below, and is IMPLEMENTATION DEFINED. For any bit that is not implemented, the associated register bit is RAZ/WI.

See also CLK_CFG0 and CLK_CFG2.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nCOLDRESETAON

Bit descriptions

Table 8-63: CLK_CFG1 bit descriptions

Bits	Name	Description	Type	Reset
[31:24]	-	Reserved	RAZ/WI	`0x00`

Bits	Name	Description	Type	Reset
[23:20]	AONCLKCFGSTATUS	Clock Configuration Status value that reports the status of clock control for AONCLK.	RO	AONCLKCFGSTATUS
[19:16]	SYSCLKCFGSTATUS	Clock Configuration Status value that reports the status of clock control for SYSCLK.	RO	SYSCLKCFGSTATUS
[15:8]	-	Reserved	RAZ/WI	`0x00`
[7:4]	AONCLKCFG	Clock Configuration value that drives AONCLKCFG signals.	RW	AONCLKCFGRST
[3:0]	SYSCLKCFG	Clock Configuration value that drives SYSCLKCFG signals.	RW	SYSCLKCFGRST

8.7.2.5 CLK_CFG2

The CLK_CFG2 register provides control register fields to drive expansion clock generation logic that drives clock for this subsystem.

Each clock control handshake comprises of a configuration request register, that is CLKCFG, and a status register, that is CLKCFGSTATUS that acts as an acknowledgment. After writing to each CLKCFG field, the software has to poll the associated CLKCFGSTATUS field until the CLKCFGSTATUS is the same as the CLKCFG before doing any other operations. In addition, if it is the first write to the register after boot, we recommend that software polls to check that the targeted CLKCFGSTATUS field matches its associated CLKCFG value before the write occurs. The actual number of bits being implemented in the related CLKCFG and CLKCFGSTATUS signals can be less than that defined below, and is IMPLEMENTATION DEFINED. For any bit that is not implemented, the associated register bit is RAZ/WI.

See also CLK_CFG0 and CLK_CFG1.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nCOLDRESETAON

Bit descriptions

Table 8-64: CLK_CFG2 bit descriptions

Bits	Name	Description	Type	Reset
[31:28]	NPU3CLKCFGSTATUS	Clock Configuration Status value that reports the status of clock control for NPU3CLK. **RAZ/WI** if NUMNPU < 4	RO	NPU3CLKCFGSTATUS
[27:24]	NPU2CLKCFGSTATUS	Clock Configuration Status value that reports the status of clock control for NPU2CLK. **RAZ/WI** if NUMNPU < 3	RO	NPU2CLKCFGSTATUS
[23:20]	NPU1CLKCFGSTATUS	Clock Configuration Status value that reports the status of clock control for NPU1CLK. **RAZ/WI** if NUMNPU < 2	RO	NPU1CLKCFGSTATUS
[19:16]	NPU0CLKCFGSTATUS	Clock Configuration Status value that reports the status of clock control for NPU0CLK. **RAZ/WI** if NUMNPU < 1	RO	NPU0CLKCFGSTATUS
[15:12]	NPU3CLKCFG	Clock Configuration value that drives NPU3CLKCFG signals. **RAZ/WI** if NUMNPU < 4	RW	NPU3CLKCFGRST
[11:8]	NPU2CLKCFG	Clock Configuration value that drives NPU2CLKCFG signals. **RAZ/WI** if NUMNPU < 3	RW	NPU2CLKCFGRST
[7:4]	NPU1CLKCFG	Clock Configuration value that drives NPU1CLKCFG signals. **RAZ/WI** if NUMNPU < 2	RW	NPU1CLKCFGRST
[3:0]	NPU0CLKCFG	Clock Configuration value that drives NPU0CLKCFG signals. **RAZ/WI** if NUMNPU < 1	RW	NPU0CLKCFGRST

8.7.2.6 CLOCK_FORCE

The Clock Force register allows software to override dynamic clock gating that may be implemented in the system and keep each clock running.

Bits 0 to 11 are clock forces that does not apply to clock gates within the design that are responsible for the gating of clocks when gating power to a power gated region. Instead, it is applied to hierarchical dynamic clock gating within the system, with one bit for each power domain. Forcing a clock ON can reduce the latency that is incurred as a result of dynamic clock control but generally has a reverse side effect of increasing the dynamic power consumption of the system. Note that all clock force default values, are set to HIGH at reset. This allows the system to boot in case any hierarchical dynamic clock control implementation is non-functional. You must clear the associated clock force register bit to enable hierarchical dynamic clock control for each power domain.

Bits 16 to 26 are clock forces that are used to force the external clock generator to continue to generate its clock. This can be useful, if desired, to avoid clock generators like PLLs from turning off, which can result in a long turn on time.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nCOLDRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-65: CLOCK_FORCE bit descriptions

Bits	Name	Description	Type	Reset
[31:27]	-	Reserved	RAZ/WI	0x00
[26]	NPU3CLK_FORCE	Set HIGH to request the input NPU3CLK source to stay ON. The field is reserved and **RAZ/WI** if NUMNPU < 4.	RW	0x0
[25]	NPU2CLK_FORCE	Set HIGH to request the input NPU2CLK source to stay ON. The field is reserved and **RAZ/WI** if NUMNPU < 3.	RW	0x0
[24]	NPU1CLK_FORCE	Set HIGH to request the input NPU1CLK source to stay ON. The field is reserved and **RAZ/WI** if NUMNPU < 2.	RW	0x0
[23]	NPU0CLK_FORCE	Set HIGH to request the input NPU0CLK source to stay ON. The field is reserved and **RAZ/WI** if NUMNPU < 1.	RW	0x0
[22]	CPU3CLK_FORCE	Set HIGH to request the input CPU3CLK source to stay ON. The field is reserved and **RAZ/WI** if NUMCPU < 3.	RW	0x0
[21]	CPU2CLK_FORCE	Set HIGH to request the input CPU2CLK source to stay ON. The field is reserved and **RAZ/WI** if NUMCPU < 2.	RW	0x0
[20]	CPU1CLK_FORCE	Set HIGH to request the input CPU1CLK source to stay ON. The field is reserved and **RAZ/WI** if NUMCPU < 1.	RW	0x0
[19]	CPU0CLK_FORCE	Set HIGH to request the input CPU0CLK source to stay ON.	RW	0x0
[18]	DEBUGCLK_FORCE	Set HIGH to request the input DEBUGCLK source to stay ON. This field is reserved and **RAZ/WI** if DEBUGCLK is not implemented.	RW	0x0
[17]	SYSCLK_FORCE	Set HIGH to request the input SYSCLK source to stay ON.	RW	0x0
[16]	AONCLK_FORCE	Set HIGH to request the input AONCLK source to stay ON.	RW	0x0
[15:12]	-	Reserved	RAZ/WI	0x0
[11]	NPU3_CLKFORCE	Set HIGH to force PD_NPU3 Local clocks to run. The field is reserved and **RAZ/WI** if NUMNPU < 4.	RW	0x1
[10]	NPU2_CLKFORCE	Set HIGH to force PD_NPU2 Local clocks to run. The field is reserved and **RAZ/WI** if NUMNPU < 3.	RW	0x1
[9]	NPU1_CLKFORCE	Set HIGH to force PD_NPU1 Local clocks to run. The field is reserved and **RAZ/WI** if NUMNPU < 2.	RW	0x1
[8]	NPU0_CLKFORCE	Set HIGH to force PD_NPU0 Local clocks to run. The field is reserved and **RAZ/WI** if NUMNPU < 1.	RW	0x1

Bits	Name	Description	Type	Reset
[7]	CPU3_CLKFORCE	Set HIGH to force PD_CPU3 Local clocks to run. The field is reserved and **RAZ/WI** if NUMCPU < 3.	RW	`0x1`
[6]	CPU2_CLKFORCE	Set HIGH to force PD_CPU2 Local clocks to run. The field is reserved and **RAZ/WI** if NUMCPU < 2.	RW	`0x1`
[5]	CPU1_CLKFORCE	Set HIGH to force PD_CPU1 Local clocks to run. The field is reserved and **RAZ/WI** if NUMCPU < 1.	RW	`0x1`
[4]	CPU0_CLKFORCE	Set HIGH to force PD_CPU0 Local clocks to run.	RW	`0x1`
[3]	-	Reserved	RAZ/WI	`0x0`
[2]	DEBUG_CLKFORCE	Set HIGH to force all clocks in PD_DEBUG to run.	RW	`0x1`
[1]	SYS_CLKFORCE	Set HIGH to force all clocks in PD_SYS to run.	RW	`0x1`
[0]	MGMT_CLKFORCE	Set HIGH to force all clocks in PD_MGMT domain to run.	RW	`0x1`

8.7.2.7 RESET_SYNDROME

The RESET_SYNDROME register stores the reason for the last Reset event. Writing HIGH to a bit results in that bit write value to be ignored and the bit maintaining its previous value. RESET_SYNDROME is cleared by software writing zero to each bit to clear. If after starting from a reset event, RESET_SYNDROME is not cleared, on another reset event, the register may no longer accurately reflect the last reset event. CPU<n>LOCKUP does not actually generate reset, but when HIGH, it indicates that a CPU has locked-up and could be a precursor to another reset event, for example, watchdog timer reset request.

The RESET_SYNDROME register always stores Reset events regardless of the COLDRESET_MODE configuration and even if the Cold reset generation is performed externally into the system through the HOSTRESETREQ reset request signal. Also note that CPU<n>LOCKUP events are always stored, and only reset requests that are not masked by their respective mask bits are stored in the register.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

The RESET_SYNDROME register resides in the PD_AON power domain but can also reside in PD_MGMT power domain if its states are saved and restored when entering and then leaving the lower power state, respectively and if its functionality is maintained while PD_MGMT is in low power state.

Reset domain

nPORESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-66: RESET_SYNDROME bit descriptions

Bits	Name	Description	Type	Reset
[31:24]	SWSYN	Software defined reset syndrome This field is set after reset caused by software setting the SWSYN field and the SWCOLDRESETREQ or SWWARMRESETREQ field in the SWRESET register. It may serve as a messaging box between software running before the reset and after the reset.	RW0C	0x0
[23:20]	-	Reserved	RAZ/WI	0x00
[19]	SWWARMRESETREQ	Software Warm Reset Request	RW0C	0x00
[18]	WARMRESETREQ	Subsystem Hardware Warm Reset Request Input If WARMRESETREQ is not implemented, this bit is reserved and **RAZ/WI**.	RW0C	0x0
[17]	SAMCRSTREQ	Security Alarm Manager Cold Reset Request. This bit is reserved and **RAZ/WI** if LCM_KMU_SAM_PRESENT = 0.	RW0C	0x0
[16]	SAMWRSTREQ	Security Alarm Manager Warm Reset Request. This bit is reserved and **RAZ/WI** if LCM_KMU_SAM_PRESENT = 0.	RW0C	0x0
[15]	CPU3LOCKUP	CPU 3 Lockup Status. This bit is reserved and **RAZ/WI** if NUMCPU < 3.	RW0C	0x0
[14]	CPU2LOCKUP	CPU 2 Lockup Status. This bit is reserved and **RAZ/WI** if NUMCPU < 2.	RW0C	0x0
[13]	CPU1LOCKUP	CPU 1 Lockup Status. This bit is reserved and **RAZ/WI** if NUMCPU < 1.	RW0C	0x0
[12]	CPU0LOCKUP	CPU 0 Lockup Status.	RW0C	0x0
[11]	CPU3RSTREQ	CPU 3 Warm Reset Request. This bit is reserved and **RAZ/WI** if NUMCPU < 3.	RW0C	0x0
[10]	CPU2RSTREQ	CPU 2 Warm Reset Request. This bit is reserved and **RAZ/WI** if NUMCPU < 2.	RW0C	0x0
[9]	CPU1RSTREQ	CPU 1 Warm Reset Request. This bit is reserved and **RAZ/WI** if NUMCPU < 1.	RW0C	0x0
[8]	CPU0RSTREQ	CPU 0 Warm Reset Request.	RW0C	0x0
[7]	HOSTRESETREQ	Host Level Cold Reset Request Input.	RW0C	0x0
[6]	LCMRSTREQ	Lifecycle Manager Secure provisioning warm reset Request. This bit is reserved and **RAZ/WI** if LCM_KMU_SAM_PRESENT = 0.	RW0C	0x0
[5]	SWCOLDRESETREQ	Software Cold Reset Request.	RW0C	0x0
[4]	RESETREQ	Subsystem Hardware Cold Reset Request Input.	RW0C	0x0
[3]	SLOWCLKWDRSTREQ	SLOWCLK Watchdog Cold Reset Request.	RW0C	0x0
[2]	SWDRSTREQ	Secure Watchdog Cold Reset Request.	RW0C	0x0
[1]	NSWDRSTREQ	Non-secure Watchdog Cold Reset Request.	RW0C	0x0
[0]	PoR	Power-On-Reset	RW0C	0x01

8.7.2.8 RESET_MASK

The RESET_MASK register allows software to control which reset sources are going to be merged to generate the system wide warm reset, nWARMRESETAON, or the nCOLDRESETAON signal. Set each bit to HIGH to enable each source. Note that each of these mask bits, if cleared, not only prevents the reset source being used to generate the reset, it also prevents the associated RESET_SYNDROME register bit from recording the event.

Depending on the COLDRESET_MODE configuration, the final reset could be generated within the subsystem, or through a higher level reset generation entity through the HOSTRESETREQ signal.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain when PILEVEL = 2 if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nWARMRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-67: RESET_MASK bit descriptions

Bits	Name	Description	Type	Reset
[31:12]	-	Reserved	RAZ/WI	`0x00_0000`
[11]	CPU3RSTREQEN	CPU 3 Warm Reset Request Enable. This bit is Reserved and **RAZ/WI** if NUMCPU < 3.	RW	CPU3RSTREQENRST
[10]	CPU2RSTREQEN	CPU 2 Warm Reset Request Enable. This bit is Reserved and **RAZ/WI** if NUMCPU < 2.	RW	CPU2RSTREQENRST
[9]	CPU1RSTREQEN	CPU 1 Warm Reset Request Enable. This bit is Reserved and **RAZ/WI** if NUMCPU < 1.	RW	CPU1RSTREQENRST
[8]	CPU0RSTREQEN	CPU 0 Warm Reset Request Enable.	RW	CPU0RSTREQENRST
[7:2]	-	Reserved	RAZ/WI	`0x00`
[1]	NSWDRSTREQEN	Non-secure Watchdog Reset Enable.	RW	`0x0`
[0]	-	Reserved	RAZ/WI	`0x0`

8.7.2.9 SWRESET

The SWRESET register allows software to request for a system Cold reset or Warm reset. To request for a reset, write '1' to the corresponding register bit. The register always returns zeros.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON or in the PD_MGMT power domain. Its content is not retained when PD_MGMT is turned off in HIBERNATION1 state when PILEVEL = 2.

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-68: SWRESET bit descriptions

Bits	Name	Description	Type	Reset
[31:24]	SWSYN	Software Reset Software defined reset syndrome. It serves as a messaging box between software running before the reset and software running after the reset. The SWSYN is written along with SWCOLDRESETREQ or SWWARMRESETREQ, otherwise it is ignored. It propagates to the RESET_SYNDROME register when the reset occurs.	RAZW1S	`0x0`
[23:20]	-	Reserved	RAZ/WI	`0x0`
[19]	SWWARMRESETREQ	Software Warm Reset Request. Write '1' to set to HIGH, to request a Warm reset.	RAZW1S	`0x0`
[18:6]	-	Reserved	RAZ/WI	`0x0000`
[5]	SWCOLDRESETREQ	Software Cold Reset Request. Write '1' to set to HIGH, to request a Cold reset.	RAZW1S	`0x0`
[4:0]	-	Reserved	RAZ/WI	`0x000`

8.7.2.10 GRETREG

The General Purpose Retention Register provides 16 bits of retention register for general storage, through HIBERANTION0 or HIBERNATION1 system power states.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

The GRETREG resides in the PD_AON power domain but can also reside in PD_MGMT power domain when PILEVEL = 2 if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nCOLDRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-69: GRETREG bit descriptions

Bits	Name	Description	Type	Reset
[31:16]	-	Reserved	RAZ/WI	`0x00000`
[15:0]	GRETREG	General Purpose Retention Register.	RW	`0x0000`

8.7.2.11 INITSVTOR<n>

The INITSVTOR<n> register is used to define the CPU <n> Initial Secure Vector table offset (VTOR_S.TBLOFF[31:7]) out of reset.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

The INITSVTOR<n> register resides in the PD_AON power domain but can also reside in PD_MGMT power domain when PILEVEL = 2 if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nCOLDRESETAON or nWARMRESETAON defined by INITSVTORRST_MODE.

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-70: INITSVTOR<n> bit descriptions

Bits	Name	Description	Type	Reset
[31:7]	INITSVTOR <n>	Default Secure Vector table offset at reset for CPU <n>.	RW	INITSVTOR<n> RST[31:7]
[6:1]	-	Reserved	RAZ/ WI	`0x00`
[0]	INITSVTOR <n> LOCK	Lock INITSVTOR<n>. When set to '1', it stops any further writes to INITSVTOR<n> and INITSVTOR<n>LOCK fields. Cleared only by warm reset.	RW1S	`0x0`

8.7.2.12 CPUWAIT

The CPUWAIT register provides controls to force each CPU to wait after reset rather than Boot Immediately. This allows another entity in the expansion system or the debugger to access the system prior to the CPU booting.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

The CPUWAIT register resides in the PD_AON power domain but can also reside in PD_MGMT power domain when PILEVEL = 2 if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nCOLDRESETAON or nWARMRESETAON defined by CPUWAITRST_MODE.

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-71: CPUWAIT bit descriptions

Bits	Name	Description	Type	Reset
[31:4]	-	Reserved	RAZ/ WI	`0x0000_0000`
[3]	CPU3WAIT	CPU 3 waits at boot. • '0': boot normally. • '1': wait at boot. When CPU3WAITCLR input is 1'b1, this bit is also cleared. This bit is Reserved and **RAZ/WI** if NUMCPU < 3.	RW	CPU3WAITRST

Bits	Name	Description	Type	Reset
[2]	CPU2WAIT	CPU 2 waits at boot. • '0': boot normally. • '1': wait at boot. When CPU2WAITCLR input is 1'b1, this bit is also cleared. This bit is Reserved and **RAZ/WI** if NUMCPU < 2.	RW	CPU2WAITRST
[1]	CPU1WAIT	CPU 1 waits at boot. • '0': boot normally. • '1': wait at boot. When CPU1WAITCLR input is 1'b1, this bit is also cleared. This bit is Reserved and **RAZ/WI** if NUMCPU < 1.	RW	CPU1WAITRST
[0]	CPU0WAIT	CPU 0 waits at boot. • '0': boot normally. • '1': wait at boot. When CPU0WAITCLR input is 1'b1, this bit is also cleared.	RW	CPU0WAITRST

8.7.2.13 NMI_ENABLE

The NMI_ENABLE register provides controls to enable or disable the internally or externally generated Non-Maskable Interrupt sources from generating an NMI interrupt on each CPU core. This allows a CPU to take control of all internal NMI interrupt sources or allow all CPUs to see the same NMI interrupts.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

The NMI_ENABLE register resides in the PD_AON power domain but can also reside in PD_MGMT power domain when PILEVEL = 2 if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

This register is reset by nWARMRESETAON and its reset value is defined by the configuration options.

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-72: NMI_ENABLE bit descriptions

Bits	Name	Description	Type	Reset
[31:20]	-	Reserved	RAZ/WI	`0x0000`
[19]	CPU3_EXPNMI_ENABLE	CPU 3 Externally Sourced NMI Enable. This determines if the input CPU3EXPNMI can raise NMI interrupt on CPU 3: • HIGH, allowed. • LOW, is masked and not allowed. This bit is reserved and **RAZ/WI** if NUMCPU < 3.	RW	CPU3EXPNMIENABLERST
[18]	CPU2_EXPNMI_ENABLE	CPU 2 Externally Sourced NMI Enable. This determines if the input CPU2EXPNMI can raise NMI interrupt on CPU 2: • HIGH, allowed. • LOW, is masked and not allowed. This bit is reserved and **RAZ/WI** if NUMCPU < 2.	RW	CPU2EXPNMIENABLERST
[17]	CPU1_EXPNMI_ENABLE	CPU 1 Externally Sourced NMI Enable. This determines if the input CPU1EXPNMI can raise NMI interrupt on CPU 1: • HIGH, allowed. • LOW, is masked and not allowed. This bit is reserved and **RAZ/WI** if NUMCPU < 1.	RW	CPU1EXPNMIENABLERST
[16]	CPU0_EXPNMI_ENABLE	CPU 0 Externally Sourced NMI Enable. This determines if the input, CPU0EXPNMI can raise NMI interrupt on CPU 0: • HIGH, allowed. • LOW, is masked and not allowed.	RW	CPU0EXPNMIENABLERST
[15:4]	-	Reserved	RAZ/WI	`0x0000`
[3]	CPU3_INTNMI_ENABLE	CPU 3 Internally Sourced NMI Enable. This determines if the subsystem internally generated NMI interrupt sources can raise NMI interrupt on CPU 3: • HIGH, allowed. • LOW, is masked and not allowed. This bit is reserved and **RAZ/WI** if NUMCPU < 3.	RW	CPU3INTNMIENABLERST
[2]	CPU2_INTNMI_ENABLE	CPU 2 Internally Sourced NMI Enable. This determines if the subsystem internally generated NMI interrupt sources can raise NMI interrupt on CPU 2: • HIGH, allowed. • LOW, is masked and not allowed. This bit is reserved and **RAZ/WI** if NUMCPU < 2.	RW	CPU2INTNMIENABLERST

Bits	Name	Description	Type	Reset
[1]	CPU1_INTNMI_ENABLE	CPU 1 Internally Sourced NMI Enable. This determines if the subsystem internally generated NMI interrupt sources can raise NMI interrupt on CPU 1: • HIGH, allowed. • LOW, is masked and not allowed. This bit is reserved and **RAZ/WI** if NUMCPU < 1.	RW	CPU1INTNMIENABLERST
[0]	CPU0_INTNMI_ENABLE	CPU 0 Internally Sourced NMI Enable. This determines if the subsystem internally generated NMI interrupt sources can raise NMI interrupt on CPU 0: • HIGH, allowed. • LOW, is masked and not allowed.	RW	CPU0INTNMIENABLERST

8.7.2.14 PPUINTSTAT

The PPU Interrupt Status Register brings together all the PPU interrupt statuses in to a single register.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain when PILEVEL = 2.

Reset domain

nWARMRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-73: PPUINTSTAT bit descriptions

Bits	Name	Description	Type	Reset
[31:12]	-	Reserved	RAZ/WI	`0x0000_0000`
[11]	DEBUG_PPU_INTSTAT	Debug PPU Interrupt	RO	`0x0`
[10]	NPU3_PPU_INTSTAT	NPU 3 PPU interrupt **RAZ/WI** if NUMNPU <4	RO	`0x0`
[9]	NPU2_PPU_INTSTAT	NPU 2 PPU interrupt **RAZ/WI** if NUMNPU <3	RO	`0x0`
[8]	NPU1_PPU_INTSTAT	NPU 1 PPU interrupt **RAZ/WI** if NUMNPU <2	RO	`0x0`

Bits	Name	Description	Type	Reset
[7]	NPU0_PPU_INTSTAT	NPU 0 PPU interrupt **RAZ/WI** if NUMNPU <1	RO	0x0
[6]	-	Reserved	RAZ/WI	0x0
[5]	CPU3_PPU_INTSTAT	CPU 3 PPU interrupt **RAZ/WI** if NUMCPU <3	RO	0x0
[4]	CPU2_PPU_INTSTAT	CPU 2 PPU interrupt **RAZ/WI** if NUMCPU <2	RO	0x0
[3]	CPU1_PPU_INTSTAT	CPU 1 PPU interrupt **RAZ/WI** if NUMCPU <1	RO	0x0
[2]	CPU0_PPU_INTSTAT	CPU 0 PPU interrupt	RO	0x0
[1]	SYS_PPU_INTSTAT	System PPU interrupt	RO	0x0
[0]	MGMT_PPU_INTSTAT	Management PPU Interrupt	RO	0x0

8.7.2.15 PWRCTRL

The Power Control register configures the power control features in the CRSAS Ma2 System.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain when PILEVEL = 2 if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nWARMRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-74: PWRCTRL bit descriptions

Bits	Name	Description	Type	Reset
[31:2]	-	Reserved	RAZ/WI	0x0000_0000
[1]	PPU_ACCESS_UNLOCK	PPU_ACCESS_FILTER write unlock. When '1', noth PPU_ACCESS_FILTER and this register bits can be written. When set to '0', the PPU_ACCESS_FILTER and this register bit is no longer writable, and PPU_ACCESS_UNLOCK stays '0'.	RW0C	0x01
[0]	PPU_ACCESS_FILTER	Filter Access to PPU Registers. When set to '1', only key PPU interrupt handling registers are open to write access, and all other PPU registers are read only. When set to '0', it releases all the PPU registers to full access. For more information in PPU registers accessibility, see Power Policy Units.	RW	0x01

8.7.2.16 PDCM_PD_SYS_SENSE

The Power Dependency Control Matrix System Power domain (PD_SYS) Sensitivity register is used to define what keeps the PD_SYS domain awake and the minimum power state to use when the domain is in its low power state.

Configurations

This register implementation depends on the configuration of individual fields.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain when PILEVEL = 2 if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nWARMRESETAON

Bit descriptions

Table 8-75: PDCM_PD_SYS_SENSE bit descriptions

Bits	Name	Description	Type	Reset
[31:30]	MIN_PWR_STATE	Defines the Minimum Power State, when PD_SYS is trying to enter a lower power state: • '00': Minimum power state is OFF, • '01': Minimum power state is Retention, • '10': Minimum power state is ON, • Others: Reserved.	RW	0x0
[29:24]	-	Reserved	RAZ/WI	0x000
[23]	S_PDCMRETQREQ3	Enable sensitivity to PDCMQRETREQn[3] signal. If set to '1', PD_SYS stays in ON or RET if PDCMRETQREQn[3] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 4.	RW	0x0
[22]	S_PDCMRETQREQ2	Enable sensitivity to PDCMQRETREQn[2] signal. If set to '1', PD_SYS stays in ON or RET if PDCMRETQREQn[2] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 3.	RW	0x0
[21]	S_PDCMRETQREQ1	Enable sensitivity to PDCMQRETREQn[1] signal. If set to '1', PD_SYS stays in ON or RET if PDCMRETQREQn[1] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 2.	RW	0x0
[20]	S_PDCMRETQREQ0	Enable sensitivity to PDCMQRETREQn[0] signal. If set to '1', PD_SYS stays in ON or RET if PDCMRETQREQn[0] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 1.	RW	0x0
[19]	S_PDCMONQREQ3	Enable sensitivity to PDCMONQREQn[3] signal. If set to '1', PD_SYS stays ON if PDCMONQREQn[3] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 4.	RW	0x0

Bits	Name	Description	Type	Reset
[18]	S_PDCMONQREQ2	Enable sensitivity to PDCMQONREQn[2] signal. If set to '1', PD_SYS stays ON if PDCMONQREQn[2] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 3.	RW	0x0
[17]	S_PDCMONQREQ1	Enable sensitivity to PDCMQONREQn[1] signal. If set to '1', PD_SYS stays ON if PDCMONQREQn[1] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 2.	RW	0x0
[16]	S_PDCMONQREQ0	Enable sensitivity to PDCMQONREQn[0] signal. If set to '1', PD_SYS stays ON if PDCMONQREQn[0] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 1.	RW	0x0
[15]	S_PD_MGMT_ON	Tied to LOW. Ignores PD_MGMT power state. This bit is reserved and **RAZ/WI** if PILEVEL < 2.	RO	0x0
[14]	-	Reserved	RAZ/WI	0x0
[13]	S_PD_DEBUG_ON	Tied to LOW. Ignores PD_DEBUG power state.	RO	0x0
[12]	-	Reserved	RAZ/WI	0x0
[11:9]	-	Reserved	RAZ/WI	0x0
[8]	S_PD_NPU3_ON	Tied to HIGH. PD_SYS always tries to stay ON if PD_NPU3 is ON. This bit is reserved and **RAZ/WI** if NUMNPU < 4.	RO	0x1
[7]	S_PD_NPU2_ON	Tied to HIGH. PD_SYS always tries to stay ON if PD_NPU2 is ON. This bit is reserved and **RAZ/WI** if NUMNPU < 3.	RO	0x1
[6]	S_PD_NPU1_ON	Tied to HIGH. PD_SYS always tries to stay ON if PD_NPU1 is ON. This bit is reserved and **RAZ/WI** if NUMNPU < 2.	RO	0x1
[5]	S_PD_NPU0_ON	Tied to HIGH. PD_SYS always tries to stay ON if PD_NPU0 is ON. This bit is reserved and **RAZ/WI** if NUMNPU < 1.	RO	0x1
[4]	S_PD_CPU3_ON	Tied to HIGH. PD_SYS always tries to stay ON if PD_CPU3 is ON. This bit is reserved and **RAZ/WI** if NUMCPU < 3 o PILEVEL < 1.	RO	0x1
[3]	S_PD_CPU2_ON	Tied to HIGH. PD_SYS always tries to stay ON if PD_CPU2 is ON. This bit is reserved and **RAZ/WI** if NUMCPU < 2 o PILEVEL < 1.	RO	0x1
[2]	S_PD_CPU1_ON	Tied to HIGH. PD_SYS always tries to stay ON if PD_CPU1 is ON. This bit is reserved and **RAZ/WI** if NUMCPU < 1 o PILEVEL < 1.	RO	0x1
[1]	S_PD_CPU0_ON	Tied to HIGH. PD_SYS always tries to stay ON if PD_CPU0 is ON. This bit is reserved and **RAZ/WI** if PILEVEL < 1.	RO	0x1
[0]	S_PD_SYS_ON	Enable PD_SYS ON Sensitivity. Set this to high to keep PD_SYS awake after powered ON.	RW	0x0

8.7.2.17 PDCM_PD_CPU<n>_SENSE

The Power Dependency Control Matrix CPU <n> Power domain (PD_CPU<n>) Sensitivity register is used to define what keeps the PD_CPU<n> domain awake.

Currently, the PD_CPU<n> are not sensitive to any incoming dependencies.

Configurations

This register implementation depends on the configuration of individual fields. When PILEVEL < 1, these registers do not exist and the register areas they occupy are Reserved, and **RAZ/WI**.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain when PILEVEL = 2 if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nWARMRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-76: PDCM_PD_CPU<n>_SENSE bit descriptions

Bits	Name	Description	Type	Reset
[31:24]	-	Reserved	RAZ/ WI	0x000
[23]	S_PDCMRETQREQ3	Tied to LOW. Ignores PDCMRETQREQn[3] signal. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 4.	RO	0x0
[22]	S_PDCMRETQREQ2	Tied to LOW. Ignores PDCMRETQREQn[2] signal. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 3.	RO	0x0
[21]	S_PDCMRETQREQ1	Tied to LOW. Ignores PDCMRETQREQn[1] signal. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 2.	RO	0x0
[20]	S_PDCMRETQREQ0	Tied to LOW. Ignores PDCMRETQREQn[0] signal. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 1.	RO	0x0
[19]	S_PDCMONQREQ3	Tied to LOW. Ignores PDCMONQREQn[3] signal. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 4.	RO	0x0
[18]	S_PDCMONQREQ2	Tied to LOW. Ignores PDCMONQREQn[2] signal. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 3.	RO	0x0
[17]	S_PDCMONQREQ1	Tied to LOW. Ignores PDCMONQREQn[1] signal. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 2.	RO	0x0
[16]	S_PDCMONQREQ0	Tied to LOW. Ignores PDCMONQREQn[0] signal. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 1.	RO	0x0
[15]	S_PD_MGMT_ON	Tied to LOW. Ignores PD_MGMT power state. This bit is reserved and **RAZ/WI** if PILEVEL < 2.	RO	0x0
[14]	-	Reserved	RAZ/ WI	0x0
[13]	S_PD_DEBUG_ON	Tied to LOW. Ignores PD_DEBUG power state.	RO	0x0
[12]	-	Reserved	RAZ/ WI	0x0
[11:5]	-	Reserved	RAZ/ WI	0x000

Bits	Name	Description	Type	Reset
[4]	S_PD_CPU3_ON	Tied to LOW. Ignores PD_CPU3 power state. This bit is reserved and **RAZ/WI** if NUMCPU < 3.	RO	`0x0`
[3]	S_PD_CPU2_ON	Tied to LOW. Ignores PD_CPU2 power state. This bit is reserved and **RAZ/WI** if NUMCPU < 2.	RO	`0x0`
[2]	S_PD_CPU1_ON	Tied to LOW. Ignores PD_CPU1 power state. This bit is reserved and **RAZ/WI** if NUMCPU < 1.	RO	`0x0`
[1]	S_PD_CPU0_ON	Tied to LOW. Ignores PD_CPU0 power state.	RO	`0x0`
[0]	S_PD_SYS_ON	Tied to LOW. Ignores PD_SYS power state.	RO	`0x0`

8.7.2.18 PDCM_PD_VMR<x>_SENSE

The Power Dependency Control Matrix Volatile Memory Region <x> Power domain (PD_VMR<x>) Sensitivity register is used to define what keeps awake the PD_VMR<x> domain and the minimum power state to use when the domain is in its low power state, where *x* is 0 to NUMVMBANK-1.

Configurations

This register implementation depends on the configuration of individual fields. When PILEVEL < 1, these registers do not exist and the register areas they occupy are Reserved, and **RAZ/WI**.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain when PILEVEL = 2 if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nWARMRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-77: PDCM_PD_VMR<x>_SENSE bit descriptions

Bits	Name	Description	Type	Reset
[31:30]	MIN_PWR_STATE	Defines the Minimum Power State, when PD_VMR<x> is trying to enter a lower power state: • ‘00’: Minimum power state is OFF, • ‘01’: Minimum power state is Retention, • ‘10’: Minimum power state is ON, • Others: Reserved.	RW	`0x01`

Bits	Name	Description	Type	Reset
[29:24]	-	Reserved	RAZ/WI	0x0
[23]	S_PDCMRETQREQ3	Enable sensitivity to PDCMRETQREQn[3] signal. If set to '1',, PD_VMR<x> stay in ON or RET if PDCMRETQREQn[3] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 4.	RW	0x0
[22]	S_PDCMRETQREQ2	Enable sensitivity to PDCMRETQREQn[2] signal. If set to '1', PD_ VMR<x> stays in ON or RET if PDCMRETQREQn[2] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 3.	RW	0x0
[21]	S_PDCMRETQREQ1	Enable sensitivity to PDCMRETQREQn[1] signal. If set to '1', PD_ VMR<x> stays in ON or RET if PDCMRETQREQn[1] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 2.	RW	0x0
[20]	S_PDCMRETQREQ0	Enable sensitivity to PDCMQRETREQn[0] signal. If set to '1', PD_ VMR<x> stays in ON or RET if [PDCMRETQREQn0] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 1.	RW	0x0
[19]	S_PDCMONQREQ3	Enable sensitivity to [PDCMONQREQn3 signal. If set to '1', D_VMR<x> stays ON if PDCMONQREQn[3] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 4.	RW	0x0
[18]	S_PDCMONQREQ2	Enable sensitivity to PDCMONQREQn[2] signal. If set to '1', PD_VMR<x> stays ON if PDCMONQREQn[2] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 3.	RW	0x0
[17]	S_PDCMONQREQ1	Enable sensitivity to PDCMONQREQn[1] signal. If set to '1', PD_VMR<x> stays ON if PDCMONQREQn[1] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 2.	RW	0x0
[16]	S_PDCMONQREQ0	Enable sensitivity to PDCMONQREQn[0] signal. If set to '1', PD_VMR<x> stays ON if PDCMONQREQn[0] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 1.	RW	0x0
[15]	S_PD_MGMT_ON	Tied to LOW. Ignores PD_MGMT power state. This bit is reserved and **RAZ/WI** if PILEVEL < 2.	RO	0x0
[14]	-	Reserved	RAZ/WI	0x0
[13]	S_PD_DEBUG_ON	Tied to LOW. Ignores PD_DEBUG power state.	RO	0x0
[12]	-	Reserved	RAZ/WI	0x0
[11:9]	-	Reserved	RAZ/WI	0x000
[8]	S_PD_NPU3_ON	Enable PD_NPU3 sensitivity. If set to '1', PD_ VMR<x> stays ON if PD_NPU3 is ON. This bit is reserved and **RAZ/WI** if NUMNPU < 4.	RW	0x0
[7]	S_PD_NPU2_ON	Enable PD_NPU2 sensitivity. If set to '1', PD_ VMR<x> stays ON if PD_NPU2 is ON. This bit is reserved and **RAZ/WI** if NUMNPU < 3.	RW	0x0
[6]	S_PD_NPU1_ON	Enable PD_NPU1 sensitivity. If set to '1' PD_ VMR<x> stays ON if PD_NPU1 is ON. This bit is reserved and **RAZ/WI** if NUMNPU < 2.	RW	0x0
[5]	S_PD_NPU0_ON	Enable PD_NPU0 sensitivity. If set to '1' PD_ VMR<x> stays ON if PD_NPU0 is ON. This bit is reserved and **RAZ/WI** if NUMNPU < 1.	RW	0x0
[4]	S_PD_CPU3_ON	Enable PD_CPU3 sensitivity. If set to '1', PD_ VMR<x> stays ON if PD_CPU3 is ON. This bit is reserved and **RAZ/WI** if NUMCPU < 3.	RW	0x0
[3]	S_PD_CPU2_ON	Enable PD_CPU2 sensitivity. If set to '1', PD_ VMR<x> stays ON if PD_CPU2 is ON. This bit is reserved and **RAZ/WI** if NUMCPU < 2.	RW	0x0
[2]	S_PD_CPU1_ON	Enable PD_CPU1 sensitivity. If set to '1' PD_ VMR<x> stays ON if PD_CPU1 is ON. This bit is reserved and **RAZ/WI** if NUMCPU < 1.	RW	0x0

Bits	Name	Description	Type	Reset
[1]	S_PD_CPU0_ON	Enable PD_CPU0 sensitivity. If set to '1' PD_ VMR<x> stays ON if PD_CPU0 is ON.	RW	0x0
[0]	S_PD_SYS_ON	Tied to LOW. Ignores PD_SYS power state.	RO	0x0

8.7.2.19 PDCM_PD_MGMT_SENSE

The Power Dependency Control Matrix PD_MGMT Power Domain Sensitivity register is used to define what keeps the PD_MGMT domains awake.

Configurations

This register implementation depends on the configuration of individual fields. When PILEVEL < 2, this register does not exist and the register area it occupies are Reserved, and **RAZ/WI**.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

nWARMRESETAON

Usage constraints

This register is Secure privileged access only. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-78: PDCM_PD_MGMT_SENSE bit descriptions

Bits	Name	Description	Type	Reset
[31:30]	MIN_PWR_STATE	Defines the Minimum Power State, when PD_MGMT is trying to enter a lower power state: • '00': Minimum power state is OFF, • Others: Reserved.	RO	0x0
[29:24]	-	Reserved	RAZ/WI	0x0
[23]	S_PDCMRETQREQ3	Enable sensitivity to PDCMRETQREQn[3] signal. If set to '1', PD_MGMT stays ON if PDCMRETQREQn[3] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 4.	RW	0x0
[22]	S_PDCMRETQREQ2	Enable sensitivity to PDCMRETQREQn[2] signal. If set to '1', PD_MGMT stays ON if PDCMRETQREQn[2] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 3.	RW	0x0
[21]	S_PDCMRETQREQ1	Enable sensitivity to PDCMRETQREQn[1] signal. If set to '1', PD_MGMT stays ON if PDCMRETQREQn[1] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 2.	RW	0x0

Bits	Name	Description	Type	Reset
[20]	S_PDCMRETQREQ0	Enable sensitivity to PDCMRETQREQn[0] signal. If set to '1', PD_MGMT stays ON if PDCMRETQREQn[0] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 1.	RW	0x0
[19]	S_PDCMONQREQ3	Enable sensitivity to PDCMONQREQn[3] signal. If set to '1', PD_MGMT stays ON if PDCMONQREQn[3] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 4.	RW	0x0
[18]	S_PDCMONQREQ2	Enable sensitivity to PDCMONQREQn[2] signal. If set to '1',PD_MGMT stays ON if PDCMONQREQn[2] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 3.	RW	0x0
[17]	S_PDCMONQREQ1	Enable sensitivity to PDCMONQREQn[1] signal. If set to '1', PD_MGMT stays ON if PDCMONQREQn[1] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 2.	RW	0x0
[16]	S_PDCMONQREQ0	Enable sensitivity to PDCMONQREQn[0] signal. If set to '1', PD_MGMT stays ON if PDCMONQREQn[0] signal is HIGH. This bit is reserved and **RAZ/WI** if PDCMQCHWIDTH < 1.	RW	0x0
[15]	S_PD_MGMT_ON	Enable sensitivity to PD_MGMT power state. If set to '1' PD_MGMT stays ON if PD_MGMT is already ON.	RW	0x01
[14]	-	Reserved	RAZ/WI	0x0
[13]	S_PD_DEBUG_ON	Tied to HIGH. PD_MGMT stays ON if PD_DEBUG power domain is ON.	RO	0x01
[12]	-	Reserved	RAZ/WI	0x0
[11:9]	-	Reserved	RO	0x000
[8]	S_PD_NPU3_ON	Tied to HIGH. PD_MGMT always tries to stay ON if PD_NPU3 is ON. This bit is reserved and **RAZ/WI** if NUMNPU < 4.	RO	0x01
[7]	S_PD_NPU2_ON	Tied to HIGH. PD_MGMT always tries to stay ON if PD_CPU2 is ON. This bit is reserved and **RAZ/WI** if NUMNPU < 3.	RO	0x01
[6]	S_PD_NPU1_ON	Tied to HIGH. PD_MGMT always tries to stay ON if PD_NPU1 is ON. This bit is reserved and **RAZ/WI** if NUMCPU < 2.	RO	0x01
[5]	S_PD_NPU0_ON	Tied to HIGH. PD_MGMT always tries to stay ON if PD_NPU0 is ON. This bit is reserved and **RAZ/WI** if NUMNPU < 1.	RO	0x01
[4]	S_PD_CPU3_ON	Tied to HIGH. PD_MGMT always tries to stay ON if PD_CPU3 is ON. This bit is reserved and **RAZ/WI** if NUMCPU < 3.	RO	0x01
[3]	S_PD_CPU2_ON	Tied to HIGH. PD_MGMT always tries to stay ON if PD_CPU2 is ON. This bit is reserved and **RAZ/WI** if NUMCPU < 2.	RO	0x01
[2]	S_PD_CPU1_ON	Tied to HIGH. PD_MGMT always tries to stay ON if PD_CPU1 is ON. This bit is reserved and **RAZ/WI** if NUMCPU < 1.	RO	0x01
[1]	S_PD_CPU0_ON	Tied to HIGH. PD_MGMT always tries to stay ON if PD_CPU0 is ON.	RO	0x01
[0]	S_PD_SYS_ON	Tied to HIGH. PD_MGMT always tries to stay ON if PD_SYS is ON.	RO	0x01

8.7.2.20 LCM_DCU_FORCE_DISABLE

The LCM DCUEN Force Disable Register can be used to override and disable the LCM debug control enable signals during boot. Bit assignment are defined in LCM Debug Control Unit.

For more information, see LCM Debug Control Unit.

The LCM_DCU_FORCE_DISABLE register is set to the inverted value of LCM_DCU_FORCE_DISABLE_INIT by hardware when the TP-Mode Lifecycle State is either in Test Chip Indication (TCI) or Virgin state. This mechanism enables the debugger with the use of nSRST to debug the complete boot flow in TCI mode. For more information on nSRST, see CPU Reset Handling.

Configurations

This register must be available when LCM_KMU_SAM_PRESENT = 1. When not available, it is **RAZ/WI**.

Attributes

Width

32-bit

Power domain

This register resides in the PD_AON power domain but can also reside in PD_MGMT power domain if its states are saved and restored when entering and then leaving the lower power state, respectively.

Reset domain

This register is reset by nCOLDRESETAON.

Usage constraints

The LCM_DCU_FORCE_DISABLE register is Secure Privileged access only and supports 32-bit RW accesses. For write access to this register, only 32-bit writes are supported. Any byte and halfword writes are ignored.

Bit descriptions

Table 8-79: LCM_DCU_FORCE_DISABLE bit descriptions

Bits	Name	Description	Type	Reset
[31:0]	FD	Allows to override LCM DCUEN signals to zero (debug disabled). These bits are associated with LCM DCUEN bits see LCM Debug Control Unit. This field is set to the inverted value of LCM_DCU_FORCE_DISABLE_INIT by hardware when the TP-Mode Lifecycle State is either in Test Chip Indication (TCI) or Virgin state.	RW	LCM_DCU_FORCE_DISABLE_INIT

8.8 CPU Private Peripheral Bus region

Each CPU, as defined by the ARMv8-M architecture specification, hosts a local Private Peripheral Bus Region (PPB) at address `0xE000_0000` to `0xE00F_FFFF`. This region is typically for integration with CoreSight debug and trace components that is normally local to each CPU and is not intended for general peripheral usage.

In the CRSAS Ma2, this region has the memory map as shown in the table below. Other than the EWIC, CPU <n> ROM Table, and the peripherals on the External PPB Expansion that is added by an integrator, the existence of all other components depends on the CPU implementation and configuration. See *Arm® Cortex®-M55 Processor Technical Reference Manual* and *Arm® Cortex®-M85 Processor Technical Reference Manual*.

Not included in the table are TPIU and the ETB, along with a system level debug CoreSight ROM table. This ROM table is also referred to as MCU ROM table in CPU Technical Reference Manual. Depending on HASCSS:

- If HASCSS = 0 where NUMCPU must be 0, the TPIU, ETB, and the MCU ROM can be integrated using the CPU<n> EPPB Interface, with the MCU ROM pointing to the TPIU, ETB, and the core's internal ROM table. We recommend that in this case, these are integrated at the following addresses:
    - TPIU at `0xE004_0000`
    - ETB at `0xE004_5000`
    - MCU ROM table at `0xE00F_E000` where an external DAP-Lite or AHB-AP point to.
- If HASCSS = 1, the TPIU and the ETB are be provided

The CRSAS Ma2 in the Shared Debug System accessible through a separate MEM-AP. The MCU ROM table can still be added to the EPPB if other debug related components local to the CPU exist so that the MCU ROM table can point to them. Note however that in such a system, the MCU ROM table does not point to the core's internal ROM. Instead, the CPU<n> ROM table provided points to the core's internal ROM. We recommend that the MCU ROM table, if it exists is integrated at `0xE00F_E000`.

When HASCSS = 1, Arm does not recommend using this region to expand the debug system, unless the intention is to implement independent debug infrastructure for each CPU. Instead the system integrator should add to the Debug APB Expansion Interface.

For more information on the ETM, CTI, PMC, SBIST and ROM table, see *Arm® Cortex®-M55 Processor Technical Reference Manual* and *Arm® Cortex®-M85 Processor Technical Reference Manual*.

Table 8-80: CPU <n> Private Peripheral Bus Region

Row ID	From address	To address	Size	Region name	Description
1	`0xE004_0000`	`0xE004_0FFF`	4KB	PPB Expansion	CPU <n> Expansion PPB Interface.
2	`0xE004_1000`	`0xE004_1FFF`	4KB	ETM[1]	Embedded Trace Module.
3	`0xE004_2000`	`0xE004_2FFF`	4KB	CTI[1]	Cross Trigger Interface.
4	`0xE004_3000`	`0xE004_4FFF`	8KB	Reserved	Reserved
5	`0xE004_5000`	`0xE004_5FFF`	4KB	PPB Expansion	CPU <n> Expansion PPB Interface.
6	`0xE004_6000`	`0xE004_6FFF`	4KB	PMC[1]	Programmable MBIST Controller.
7	`0xE004_7000`	`0xE004_7FFF`	4KB	EWIC	External Wakeup Interrupt Controller. See EWIC.
8	`0xE004_8000`	`0xE004_8FFF`	4KB	PPB Expansion	Reserved for SBIST - Software Based Build In Self Test. Routed to CPU<n> Expansion PPB Interface on PD_CPU<n> power domain.
9	`0xE004_9000`	`0xE00F_EFFF`	24KB	PPB Expansion	CPU <n> Expansion PPB Interface.

Row ID	From address	To address	Size	Region name	Description
10	`0xE00F_F000`	`0xE00F_FFFF`	4KB	ROM table	CPU <n> Internal ROM table.

[1] The existence of the following components is dependent of the configuration and the supported features of the CPU integrated. If they do not exist, these regions are reserved and **RAZ/WI** response when accessed.

- ETM
- CTI
- PMC

8.8.1 EWIC

The CRSAS Ma2 is designed to always support an External Wakeup Interrupt Controller (EWIC) for each CPU in the system. This allows the system to support each CPU being switched off independently, and for the EWIC to run at a lower clock rate to help reduce PD_AON dynamic and leakage power consumption.

All EWICs reside in the PD_AON power domain, run on AONCLK, and reside in the nWARMRESETAON reset domain.

Each EWIC <n> is only accessible through the Private Peripheral Bus Region that is associated with CPU <n> at address `0xE004_7000` to `0xE004_7FFF`.

For more information, see *Arm® Cortex®-M55 Processor Technical Reference Manual* and *Arm® Cortex®-M85 Processor Technical Reference Manual*.

8.9 Debug System Access Region

The CRSAS Ma2 supports two key configuration options for the debug system.

The supported configuration options for the debug system are as follows:

- HASCSS = 0. CoreSight SoC-600M based Debug System does not exist. See CoreSight SoC-600M based Debug System not implemented.
- HASCSS = 1. CoreSight SoC-600M based Debug System exists. See CoreSight SoC-600M based Debug System implemented.

8.9.1 CoreSight SoC-600M based Debug System not implemented

The HASCSS parameter lets you define if the CoreSight SoC-600M based Debug System is implemented.

The value of the HASCSS parameter is as follows:

- CoreSight SoC-600M based Debug System does not exist, HASCSS = 0.
- CoreSight SoC-600M based Debug System exists, HASCSS = 1.

When the Common CoreSight Debug Infrastructure does not exist, the Debug System Access Region is not used and therefore the following regions are reserved. These regions when accessed return a bus error response.

- `0xE010_0000` to `0xE01F_FFFF`
- `0xF010_0000` to `0xF01F_FFFF`

Without the CoreSight based Debug System, All Debug related interfaces of each processor core are made available as expansion interfaces. A system integrator has to create a debug infrastructure to provide debug access to each core. This infrastructure has to use the debug authentication and debug access control signals to control accessibility to the cores and the system. For more information on these signals, see Debug Authentication interface.

8.9.2 CoreSight SoC-600M based Debug System implemented

When the CoreSight based System Debug infrastructure exists within an implementation of the CRSAS Ma2 (HASCSS=1), the system provides several Memory Access Ports (MEM-APs) to provide access to each CPU and to the shared debug components in the system.

These ports are mapped to the Debug system region with a relative address map as shown in the following table.

Table 8-81: Debug system address map

Row ID	From address	To adress	Size	Region name	Description
1	`0x0000`	`0x0FFF`	4KB	DSROM	Debug System ROM.
2	`0x1000`	`0x1FFF`	4KB	Reserved	Reserved
3	`0x2000`	`0x3FFF`	8KB	SYS APB-AP	Shared Debug System Memory Access Port.
4	`0x4000`	`0x5FFF`	8KB	CPU0 AHB-AP	CPU 0 Debug System Memory Access Port.
5	`0x6000`	`0x7FFF`	8KB	CPU1 AHB-AP	CPU 1 Debug System Memory Access Port. This AP does not exist when NUMCPU < 1, and the region is Reserved
6	`0x8000`	`0x9FFF`	8KB	CPU2 AHB-AP	CPU 2 Debug System Memory Access Port. This AP does not exist when NUMCPU < 2, and the region is Reserved
7	`0xA000`	`0xBFFF`	8KB	CPU3 AHB-AP	CPU 3 Debug System Memory Access Port. This AP does not exist when NUMCPU < 3, and the region is Reserved

Row ID	From address	To adress	Size	Region name	Description
8	`0xC000`	`0xF_FFFF`		Reserved	Reserved

This region is accessible through:

- The Debug Access Interface with an address offset of `0x0000`. Access is first controlled using the Debug Authentication Access Control signal DAPDSSACCEN. When DAPDSSACCEN = 0, all memory access port's enable signals associated with external debugger accesses, ap_en0 and ap_secure_en0, are pulled low preventing the external debugger from accessing both the CPUs and the Shared Debug System. If DAPDSSACCEN = 1, then all memory access port enable signals associated with external debugger accesses, ap_en0 and ap_secure_en0 are driven by the usual combined version of the debug authentication signals DBGEN, NIDEN, SPIDEN and SPNIDEN. If each memory access port only has support for one set of ap_en0 and ap_secure_en0, then the access from the Debug Access Interface must then be gated using the DAPDSSACCEN so that when DAPDSSACCEN = 0, accesses from the Debug Access Interface are blocked and return error responses. Arm recommends that a separate set of ap_en0 and ap_secure_en0 is used.

  A additional DAPACCEN is provided to optionally allow the integrator to control a DAP interface directly to stop external debugger access even reaching the Debug Access Interface in the first place. This allows debug components in the above memory map, with DAPACCEN = 1, to still be accessible to the external debugger while DAPDSSACCEN = 0, but yet blocks all accesses being initied from all memory access ports.

- The Main and Peripheral Interconnect are at the following two aliased address offsets:
  - `0xE010_0000` is the Non-secure alias
  - `0xF010_0000` is the Secure alias

  Access from the interconnect is gated by the Debug Authentication Access Control signal, SYSDSSACCEN<n>. When SYSDSSACCEN<n> = 0, accesses from CPU <n> through the Main Interconnect are blocked and return error responses. If SYSDSSACCEN<n> = 1, accesses from CPU <n> are allowed to pass through.

  When access can pass the first gate controlled by SYSDSSACCEN<n> into the Debug System, a PPC0 is then used to perform the final mapping of all APs either to the Non-secure region from `0xE010_0000` to `0xE01F_FFFF` or to the Secure region from `0xF010_0000` to `0xF01F_FFFF`. For more information, see PERIPHNSPPC0, PERIPHNSPPC1, PERIPHSPPPC0, PERIPHSPPPC1, PERIPHNSPPPC0, and PERIPHNSPPPC1.

  When the access finally arrives at the memory access port, then all memory access port enable signals associated with internal system access, are driven by the usual combined version of the debug authentication signals DBGEN, NIDEN, SPIDEN and SPNIDEN to control debug accessibility.

Each Memory Access Port (MEM-AP) is a twin MEM-AP that enables an external debugger to use one logical MEM-AP, and on-chip software to use a separate logical MEM-AP. A twin MEM-AP consists of two sets of 4K registers. The bottom 4K is for the external debugger accesses exclusively while the top 4K is for on-chip software accesses exclusively. It is **IMPLEMENTATION DEFINED** if the external debugger only has access to the top 4K and vice versa if the on-chip software only has access to the bottom 4K.

The following sections list the memory map behind each MEM-AP and the contents of CoreSight Debug ROMs.

8.9.2.1 Shared debug system MEM-AP memory map

The Shared debug system MEM-AP provides access to the Shared debug system that all CPUs in the system shares. It includes a trace funnel for funneling all trace data together, an Embedded Trace Buffer (ETB) that allows trace data to be stored and read by a debugger. In addition, it provides access to debug components that reside in the expansion system through the Debug APB Expansion Interface.

The MEM-AP's base address static configuration is configured to point to Shared debug system CoreSight ROM table at address `0x0000_0000`.

The following table shows the memory map that is visible to the Shared Debug System MEM-AP.

Table 8-82: Shared debug system MEM-AP memory map

Row ID	From address	To address	Size	Region name	Description
1	`0x0000_0000`	`0x0000_0FFF`	4KB	SDSROM	Shared Debug System ROM table.
2	`0x0000_1000`	`0x0000_1FFF`	4KB	SDSFUNNEL	Shared Debug System Trace Funnel.
3	`0x0000_2000`	`0x0000_2FFF`	4KB	SDSCTI	Shared Debug System Cross Trigger Interface.
4	`0x0000_3000`	`0x0000_3FFF`	4KB	SDSETB	Shared Debug System Embedded Trace Buffer.
5	`0x0000_4000`	`0x0000_FFFF`	-	Reserved	Reserved
6	`0x0001_0000`	`0xFFFF_FFFF`	-	Debug APB Expansion Interface	Debug APB Expansion Interface Region.

8.9.2.2 CPU<n> debug system MEM-AP memory map

The CPU<n> Debug System Memory Access Port provides access to the CPU<n> Debug CoreSight ROM table and the CPU Debug access interface

CPU<n> Debug CoreSight ROM table

The base address static configuration of the CPU<n> MEM-AP is configured to point to here. This ROM table includes Granular Power Requester (GPR) functionality to allow software to wake CPU<n>.

CPU Debug access interface

This provides access to the processor control and debug logic, and to a view of memory which is consistent with that observed by CPU<n>. This includes the CPU's PPB region, where other CPU<n> private debug components can be hosted. For more information, see CPU Private Peripheral Bus region and *Arm® Cortex®-M55 Processor Technical Reference Manual* and *Arm® Cortex®-M85 Processor Technical Reference Manual*.

The following table shows the memory map of the memory map as seen by the CPU<n> MEM-AP.

Table 8-83: CPU<n> MEM-AP memory map

Row ID	From address	To address	Size	Region name	Description
1	`0x0000_0000`	`0xF000_7FFF`	-	SYSACC	System Memory Access through CPU<n> Debug Access Interface.
2	`0xF000_8000`	`0xF000_8FFF`	4KB	CPU<n>ROM	CPU<n> Debug ROM table.
3	`0xF000_9000`	`0xF000_9FFF`	4KB	Reserved	Reserved
4	`0xF000_A000`	`0xFFFF_FFFF`	-	SYSACC	System Memory Access through CPU<n> Debug Access Interface.

8.9.2.3 DSROM, Debug System ROM

The Debug System ROM is a CoreSight Class `0x9` ROM table that provides a list of pointers to Access Port within the Debug System.

The following table lists the contents of the Debug System ROM table. For more details, refer to the css600_apbrom device in *Arm® CoreSight™ System-on-Chip SoC-600M Technical Reference Manual*.

Table 8-84: Debug System CoreSight ROM table contents

Offset	Name	Type	Reset	Width	Description
`0x000`	ROMENTRY0	RO	`0x0000_2001`	32-bit	ROM entry pointing to the Shared Debug System MEM-AP.
`0x004`	ROMENTRY1	RO	`0x0000_4001`	32-bit	ROM entry pointing to Debug System CPU0 MEM-AP.
`0x008`	ROMENTRY2	RO	`0x0000_6001`	32-bit	ROM entry pointing to CPU1 Debug System MEM-AP. This register is reserved and **RAZ/WI** if NUMCPU < 1.
`0x00C`	ROMENTRY3	RO	`0x0000_8001`	32-bit	ROM entry pointing to CPU2 Debug System MEM-AP. This register is reserved and **RAZ/WI** if NUMCPU < 2.
`0x010`	ROMENTRY4	RO	`0x0000_A001`	32-bit	ROM entry pointing to CPU3 Debug System MEM-AP. This register is reserved and **RAZ/WI** if NUMCPU < 3.
`0x014` – `0xFB4`	-	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xFB8`	AUTHSTATUS	RO	`0x0000_0000`	32-bit	Authentication Status Register.
`0xFBC`	DEVARCH	RO	`0x4770_0AF7`	32-bit	Device Architecture Register.
`0xFC0` – `0xFC4`	Reserved	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xFC8`	DEVID	RO	`0x0000_0000`	32-bit	Device Configuration Register.
`0xFCC`	Reserved	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xFD0`	PIDR4	RO	`0x0000_0004`	32-bit	Peripheral ID 4
`0xFD4` – `0xFDC`	-	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xFE0`	PIDR0	RO	`0x0000_004B`	32-bit	Peripheral ID 0: PIDR0[7:0] - Part Number bits [7:0].

Offset	Name	Type	Reset	Width	Description
`0xFE4`	PIDR1	RO	`0x0000_00B7`	32-bit	Peripheral ID 1: PIDR1[3:0] - Part Number [11:8], PIDR1[7:4] - JEP106 Identity Code [3:0].
`0xFE8`	PIDR2	RO	`0x0000_000B`	32-bit	Peripheral ID 2: PIDR2[2:0] - JEP106 Identity Code [6:4], PIDR2[3] - JEDEC identifier, PIDR2[7:4] - Revision Code.
`0xFEC`	PIDR3	RO	`0x0000_0000`	32-bit	Peripheral ID 3
`0xFF0`	CIDR0	RO	`0x0000_000D`	32-bit	Component ID 0
`0xFF4`	CIDR1	RO	`0x0000_0090`	32-bit	Component ID 1
`0xFF8`	CIDR2	RO	`0x0000_0005`	32-bit	Component ID 2
`0xFFC`	CIDR3	RO	`0x0000_00B1`	32-bit	Component ID 3

8.9.2.4 SDSROM, Shared Debug System ROM

The Shared Debug System Debug CoreSight ROM is a CoreSight Class `0x9` ROM table that provides a list of pointers.

The table provides pointers to the following:

- CoreSight components within the Shared Debug System level
- An external CoreSight ROM through the Debug APB Expansion interface

The following table lists the contents of the Debug System CoreSight ROM. For more details, refer to the css600_apbrom device in *Arm® CoreSight™ System-on-Chip SoC-600M Technical Reference Manual*.

Table 8-85: Debug System ROM table contents

Offset	Name	Type	Reset	Width	Description
`0x000`	ROMENTRY0	RO	`0x0000_1001`	32-bit	ROM entry pointing to the trace funnel.
`0x004`	ROMENTRY1	RO	`0x0000_2001`	32-bit	ROM entry pointing to the Cross Trigger Interface.
`0x008`	ROMENTRY2	RO	`0x0000_3001`	32-bit	ROM entry pointing to the Embedded Trace Buffer.
`0x00C`	ROMENTRY3	RO	`0x0000_4001`	32-bit	ROM entry pointing to an external CoreSight ROM table at address `0x0008_0000` through the Debug APB Expansion Interface.
`0x010` – `0xFB4`	Reserved	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xFB8`	AUTHSTATUS	RO	`0x0000_0000`	32-bit	Authentication Status Register.
`0xFBC`	DEVARCH	RO	`0x4770_0AF7`	32-bit	Device Architecture Register.
`0xFC0` – `0xFC4`	Reserved	RAZ/WI	`0x0000_0000`	32-bit	Reserved

Offset	Name	Type	Reset	Width	Description
`0xFC8`	DEVID	RO	`0x0000_0000`	32-bit	Device Configuration Register.
`0xFCC`	Reserved	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xFD0`	PIDR4	RO	`0x0000_0004`	32-bit	Peripheral ID 4
`0xFD4` – `0xFDC`	Reserved	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xFE0`	PIDR0	RO	`0x0000_004C`	32-bit	Peripheral ID 0: PIDR0[7:0] - Part Number bits [7:0].
`0xFE4`	PIDR1	RO	`0x0000_00B7`	32-bit	Peripheral ID 1: PIDR1[3:0] - Part Number [11:8], PIDR1[7:4] - JEP106 Identity Code [3:0].
`0xFE8`	PIDR2	RO	`0x0000_000B`	32-bit	Peripheral ID 2: PIDR2[2:0] - JEP106 Identity Code [6:4], PIDR2[3] - JEDEC identifier, PIDR2[7:4] - Revision Code.
`0xFEC`	PIDR3	RO	`0x0000_0000`	32-bit	Peripheral ID 3
`0xFF0`	CIDR0	RO	`0x0000_000D`	32-bit	Component ID 0
`0xFF4`	CIDR1	RO	`0x0000_0090`	32-bit	Component ID 1
`0xFF8`	CIDR2	RO	`0x0000_0005`	32-bit	Component ID 2
`0xFFC`	CIDR3	RO	`0x0000_00B1`	32-bit	Component ID 3

8.9.2.5 CPU<n>ROM, CPU<n> Debug System ROM

The CPU<n> Debug System ROM is a CoreSight Class `0x9` ROM table that provides a list of pointers.

The table provides pointers to the following:

- CoreSight ROM in the processor core.
- An optional CoreSight ROM, call the MCUROM residing on the EPPB. It also provides GPR functional to allow a debugger to request the processor to turn on.

The following table lists the contents of the CPU<n> Debug ROM. For more details, refer to the css600_apbrom_gpr device in *Arm® CoreSight™ System-on-Chip SoC-600M Technical Reference Manual*.

Table 8-86: CPU<n> Debug System ROM table contents

Offset	Name	Type	Reset	Width	Description
`0x000`	ROMENTRY0	RO	`0xE00F_F007`	32-bit	ROM entry pointing to the CPU's internal CoreSight ROM table and provides a POWERID value of `0x0`.

Offset	Name	Type	Reset	Width	Description
`0x004`	ROMENTRY1	RO	CFG_DEF	32-bit	ROM entry pointing to the MCU ROM CoreSight Debug ROM table: ROMENTRY1[31:12] =CPU<n> MCUROMADDR[31:12], ROMENTRY1[11:9] = `0x0`, ROMENTRY1[8:4] = POWERID = `0x0`, ROMENTRY[3] = `0b0`, ROMENTRY[2] = `0b1`, ROMENTRY[1:0] = {CPU<n>MCUROMVALID, CPU<n>MCU ROMVALID}.
`0x010` – `0x9FC`	-	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xA00`	DBGPCR0	RW	`0x0000_0001`	32-bit	Debug Power Control Register 0. For requesting the CPU to turn ON.
`0xA04` – `0xA7C`	-	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xA80`	DBGPSR0	RO	`0x0000_0000`	32-bit	Debug Power Status Register 0
`0xA84` – `0xBFC`	-	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xC00`	PRIDR0	RO	`0x0000_0001`	32-bit	Power Request Identification Register
`0xC04` – `0xC0C`	-	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xC10`	DBGRSTRR	RW	`0x0000_0000`	32-bit	Debug Reset Request Register. This register is not used in the CRSAS Ma2.
`0xC14`	DBGRSTAR	RO	`0x0000_0000`	32-bit	Debug Reset Acknowledge Register. This register and will always reads as zeros.
`0xC18`	SYSRSTRR	RW	`0x0000_0000`	32-bit	System Reset Request Register. This register is not used in the CRSAS Ma2.
`0xC1C`	SYSRSTAR	RO	`0x0000_0000`	32-bit	System Reset Acknowledge Register. This register and will always reads as zeros.
`0xC20` – `0xFB4`	-	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xFB8`	AUTHSTATUS	RO	`0x0000_0000`	32-bit	Authentication Status Register.
`0xFBC`	DEVARCH	RO	`0x4770_0AF7`	32-bit	Device Architecture Register.
`0xFC0` – `0xFC4`	Reserved	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xFC8`	DEVID	RO	`0x0000_0030`	32-bit	Device ID Registers
`0xFCC`	DEVTYPE	RO	`0x0000_0000`	32-bit	DEVTYPE
`0xFD0`	PIDR4	RO	`0x0000_0004`	32-bit	Peripheral ID 4
`0xFD4` – `0xFDC`	-	RAZ/WI	`0x0000_0000`	32-bit	Reserved
`0xFE0`	PIDR0	RO	`0x0000_004D`	32-bit	Peripheral ID 0: PIDR0[7:0] - Part Number bits [7:0].

Offset	Name	Type	Reset	Width	Description
`0xFE4`	PIDR1	RO	`0x0000_00B7`	32-bit	Peripheral ID: PIDR1[3:0] - Part Number [11:8], PIDR1[7:4] - JEP106 Identity Code [3:0].
`0xFE8`	PIDR2	RO	`0x0000_000B`	32-bit	Peripheral ID 2: PIDR2[2:0] - JEP106 Identity Code [6:4], PIDR2[3] - JEDEC identifier, PIDR2[7:4] - Revision Code.
`0xFEC`	PIDR3	RO	`0x0000_0000`	32-bit	Peripheral ID 3
`0xFF0`	CIDR0	RO	`0x0000_000D`	32-bit	Component ID 0
`0xFF4`	CIDR1	RO	`0x0000_0090`	32-bit	Component ID 1
`0xFF8`	CIDR2	RO	`0x0000_0005`	32-bit	Component ID 2
`0xFFC`	CIDR3	RO	`0x0000_00B1`	32-bit	Component ID 3

8.10 System boot flow

The CRSAS Ma2 defines a recommended boot flow for CPU0 that is expected to be in ROM code and executed before First Stage Boot Loader (FSBL). If implemented, CPU{1-3} could be booted in different ways using CPUWAIT and INITSVTOR<n> registers.

The following configuration is an example of booting CPU{1-3} using CPUWAIT and INITSVTOR<n> registers:

Required configurations for the system boot flow

- LCM, SAM, and KMU must exist:
  - LCM_KMU_SAM_PRESENT = 1
- NMI are routed to CPU0
  - CPU0EXPNMIENABLERST = 1 and CPU0INTNMIENABLERST = 1
  - CPU1EXPNMIENABLERST = 0 and CPU1INTNMIENABLERST = 0
  - CPU2EXPNMIENABLERST = 0 and CPU2INTNMIENABLERST = 0
  - CPU3EXPNMIENABLERST = 0 and CPU3INTNMIENABLERST = 0
- Boot from the same reset vector on all processors
  - INITSVTOR0RST[31:7] = INITSVTOR1RST[31:7] = INITSVTOR2RST[31:7] = INITSVTOR3RST[31:7]
- Start all processors after power on
  - CPU<n>WAITRST = 0

- Disable all debug features that can be used to access the RAMs or disturb Secure asset provisioning using the LCM_DCU_FORCE_DISABLE_INIT configuration option

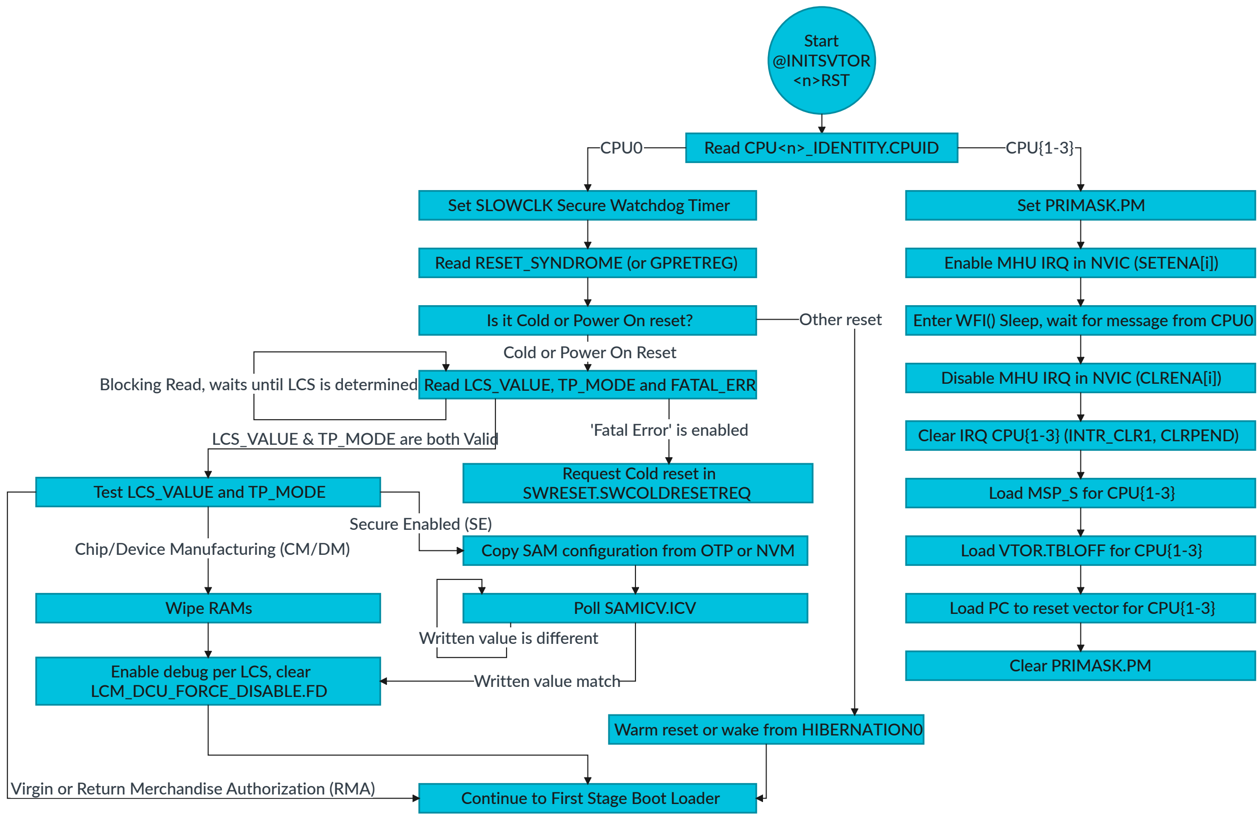
Boot flow features

- Simultaneous boot of all processors when all processors boot from the same ROM at the same time in parallel.
    - The dedicated CPU0 executes ROM code while the other processors are in sleep state.
- Secure provisioning of confidential assets into OTP in an untrusted environment when life-cycle state is either LCM Chip Manufacturing (CM) or LCM Device Manufacturing (DM).
    - The provisioning of confidential assets is possible using the LCM Secure Provisioning feature.
    - The RAMs which may hold the confidential assets are wiped immediately after Cold reset. For more information, see *Arm® Lifecycle Manager (LCM) Specification*.
- Security Alarm Manager (SAM) configuration. Load the SAM configuration data from the LCM-managed OTP array into the SAM registers.
    - This process must complete successfully before continuing with the rest of the First Stage Boot Loader process.
- Support allowing debug access from reset using the LCM_DCU_FORCE_DISABLE_INIT configuration when the TP-Mode Lifecycle State is either in Test Chip Indication (TCI) or Virgin state.

Boot steps

The following diagram shows an example of the boot flow.

Figure 8-1: Boot flow example


Start @INITSVTOR <n>RST
Read CPU<n>_IDENTITY.CPUID
CPU0
CPU{1-3}
Set SLOWCLK Secure Watchdog Timer
Read RESET_SYNDROME (or GPRETREG)
Is it Cold or Power On reset?
Other reset
Cold or Power On Reset
Blocking Read, waits until LCS is determined
Read LCS_VALUE, TP_MODE and FATAL_ERR
LCS_VALUE & TP_MODE are both Valid
'Fatal Error' is enabled
Request Cold reset in SWRESET.SWCOLDRESETREQ
Test LCS_VALUE and TP_MODE
Secure Enabled (SE)
Chip/Device Manufacturing (CM/DM)
Copy SAM configuration from OTP or NVM
Wipe RAMs
Poll SAMICV.ICV
Written value is different
Enable debug per LCS, clear LCM_DCU_FORCE_DISABLE.FD
Written value match
Warm reset or wake from HIBERNATION0
Virgin or Return Merchandise Authorization (RMA)
Continue to First Stage Boot Loader
Set PRIMASK.PM
Enable MHU IRQ in NVIC (SETENA[i])
Enter WFI() Sleep, wait for message from CPU0
Disable MHU IRQ in NVIC (CLRENA[i])
Clear IRQ CPU{1-3} (INTR_CLR1, CLRPEND)
Load MSP_S for CPU{1-3}
Load VTOR.TBLOFF for CPU{1-3}
Load PC to reset vector for CPU{1-3}
Clear PRIMASK.PM

When debugging codes for CPU{1-3}, unless using “connect only” debug setup, make sure the processor starts up from the correct reset handler so that the PC and the stacks are initialised using this flow.

8.11 Secure Asset Provisioning Flow

The LCM provides support for Secure provisioning of confidential assets into OTP in an untrusted environment.

For more information on the LCM, see the Lifecycle Manager.

The CRSAS Ma2 outlines an example provisioning flow for CPU0. This example provisioning flow is expected to be in ROM code and executed as the first part of the First Stage Boot Loader (FSBL) in provisioning mode. CPU{1-3} are not involved and have to be in sleep or wait state during provisioning.

Prerequisites for the provisioning flow

- LCM must be configured
    - LCM_KMU_SAM_PRESENT = 1

- The Lifecycle State must be either
  - CM Chip Manufacture
  - DM Device Manufacture

Provisioning flow steps

Software completes this flow in ROM but it may be initiated by the debugger.

1. While in CM or DM states, the encrypted asset and provisioning code must be placed in one of the system memories that was previously wiped by the ROM code. This may be inserted through debug interface or any other functional IOs
2. Software or debugger writes `0x2A_AAAA` to LCM DCUEN[21:0] register

   Skipping this step would result in signaling alarm in step 4 when the DCU_SP_DISABLE_MASK_VAL is applied on DCUEN.
3. LCM changes DCUEN[21:0] signals to `0x2A_AAAA`.
4. The Debug Authentication interface signals are disabled to prevent debug access.
5. Software enables Secure Provisioning mode in SP_ENABLE register of LCM to initiate Secure provisioning process.
6. The LCM 'changes' the DCUEN[21:0] per the DCU_SP_DISABLE_MASK_VAL (which is `0x2A_AAAA`) and no change on DCUEN[21:0] is expected here.
7. LCMRSTREQ requests system Warm reset mode, see RESET_SYNDROME and Reset infrastructure.
8. CPU boots again. Debug is force disabled by DCUEN[21:0].
9. ROM code can authenticate and decrypt secret assets and write to OTP with debug disabled.
10. After completion of the write to OTP, software must perform a cold reset, by setting the SWCOLDRESETREQ field of the SWRESET register.

Note

LCM DCU_EN{0-3}[127:22] register bits and DCUEN[127:22] signals are not mentioned here, but the written values and the resultant signals must take care of the per SoC use of DCUEN.

After last step above, contents of memory that may be used to store decrypted assets may not be wiped. To prevent content from inadvertently being made available to the non-secure world, only memory never to be accessed by the non-secure world is to be used for this storage. Alternatively, when the firmware boots at least for the first time, the boot code takes the earliest possible time to wipe that memory prior to continuing the boot process and enabling debug access if required. It is insuffcient to depend on the provisioning code to wipe that memory since the wiping process could be interrupted.

Appendix A System timer components

The system timer 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 behaviour causing reset of the system if a period elapses without intervention.

The components are software-programmable using APB interfaces.

A.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).

A.2 Counter operation flows

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

Counter initialization

To initialise 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.

A.3 System counter Programmers model

This section describes the 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, so 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.

A.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 A-1: CNTControlBase Frame Registers (System Counter)

Offset	Name	Type	Reset value	Description
`0x000`	CNTCR	RW	`0x0000_0000`	Counter Control Register. See CNTCR, Counter Control register
`0x004`	CNTSR	RO	`0x0000_000X`	Counter Status Register. See CNTSR, Counter Status register
`0x008`	CNTCV[31:0]	RW	`0xXXXX_XXXX`	Counter Count Value register. See CNTCV, Counter Count Value register
`0x010`	CNTSCR	RW	`0x0100_0000`	Counter Counter Scale register. See CNTSCR, Counter Scale register
`0x014-0x018`	-	RAZ/WI	`0x0000_0000`	Reserved
`0x01C`	CNTID	RO	`0x000X_000X`	Counter ID register. See CNTID, Counter ID register
`0x020-0x0BC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0x0C0-0x0CC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0x0D0`	CNTSR0	RW	`0x010_0000`	Counter Scale Register 0. See CNTSCR0, CNTSCR1, Counter Scaling registers
`0x0D4`	CNTSR1	RW	`0x010_0000`	Counter Scale Register 1. See CNTSCR0, CNTSCR1, Counter Scaling registers
`0x0D8-0xFCC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFD0`	CNTPIDR4	RO	`0x0000_0004`	Peripheral Identification Register 4.
`0xFD4-0xFDC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFE0`	CNTPIDR0	RO	`0x0000_00BA`	Peripheral Identification Register 0.
`0xFE4`	CNTPDR1	RO	`0x0000_00B0`	Peripheral Identification Register 1.
`0xFE8`	CNTPIDR2	RO	`0x0000_000B`	Peripheral Identification Register 2.
`0xFEC`	CNTPIDR3	RO	`0x0000_0000`	Peripheral Identification Register 3.
`0xFF0`	CNTCIDR0	RO	`0x0000_000D`	Component Identification Register 0.
`0xFF4`	CNTCIDR1	RO	`0x0000_00F0`	Component Identification Register 1.
`0xFF8`	CNTCIDR2	RO	`0x0000_0005`	Component Identification Register 2.
`0xFFC`	CNTCIDR3	RO	`0x0000_00B1`	Component Identification Register 3.

A.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 A-2: CNTReadBase Frame Registers (System Counter)

Offset	Name	Type	Reset value	Description
`0x000`	CNTCV[31:0]	RO	`0xXXXX_XXXX`	Counter Count Value register. See CNTCV, Counter Count Value register
`0x004`	CNTCV[63:32]	RO	`0xXXXX_XXXX`	Counter Count Value register. See CNTCV, Counter Count Value register
`0x008-0xFCC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFD0`	CNTPIDR4	RO	`0x0000_0004`	Peripheral Identification Register 4.
`0xFD4-0xFDC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFE0`	CNTPIDR0	RO	`0x0000_00BB`	Peripheral Identification Register 0.
`0xFE4`	CNTPIDR1	RO	`0x0000_00B0`	Peripheral Identification Register 1.
`0xFE8`	CNTPIDR2	RO	`0x0000_000B`	Peripheral Identification Register 2.
`0xFEC`	CNTPIDR3	RO	`0x0000_0000`	Peripheral Identification Register 3.

Offset	Name	Type	Reset value	Description
`0xFF0`	CNTCIDR0	RO	`0x0000_000D`	Component Identification Register 0.
`0xFF4`	CNTCIDR1	RO	`0x0000_00F0`	Component Identification Register 1.
`0xFF8`	CNTCIDR2	RO	`0x0000_0005`	Component Identification Register 2.
`0xFFC`	CNTCIDR3	RO	`0x0000_00B1`	Component Identification Register 3.

A.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.

A.3.4 CNTCR, Counter Control register

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

The following table shows the bit assignments.

Table A-3: CNTCR register bit assignments

Bits	Name	Type	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 CNTSCR0, CNTSCR1, Counter Scaling registers
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.

Bits	Name	Type	Reset value	Function
0	EN	RW	`0x0`	Enable Counter: • 0: Disabled: Count is not incrementing. • 1: Enabled: Count is incrementing.

A.3.5 CNTSR, Counter Status register

The CNTSR register provides Counter frequency status information.

The following table shows the bit assignments.

Table A-4: CNTSR register bit assignments

Bits	Name	Type	Reset value	Function
31:2	-	RAZ/WI	`0x000000`	Reserved
1	DBGH	RO	Unknown	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

A.3.6 CNTCV, Counter Count Value register

The CNTCV register indicates the current count value.

The following table shows the bit assignments.

Table A-5: CNTCV register bit assignments

Bits	Name	Type	Function
63:0	CountValue	• RO from CNT ReadBase • RW from CNT ControlBase	Indicates the count value.

A.3.7 CNTSCR, Counter Scale register

The CNTSCR registers store the Counter Scaling value.

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

The following table shows the bit assignments.

Table A-6: CNTSCR register bit assignments

Bits	Name	Type	Reset value	Function
31:0	ScaleVal	RW	`0x0100_0000`	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.

Note

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

A.3.8 CNTID, Counter ID register

The CNTID register indicates additional information about Counter Scaling implementation.

The following table shows the bit assignments.

Table A-7: CNTID register bit assignments

Bits	Name	Type	Reset value	Function
31:20	-	RAZ/WI	`0x0000`	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:

Bits	Name	Type	Reset value	Function
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: • 0: Invalid status, Counter not incrementing. • 1: CLK0 (REFCLK). ```<br>``` • 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

A.3.9 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 it is not necessary to program the scaling increment value each time when the 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 hardware 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 because of the Counter disabling.

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

The following table shows the bit assignments.

Table A-8: CNTSCR* register bit assignments

Bits	Name N	Type	Reset value	Function
31:0	ScaleVal	RW	`0x0100_0000`	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, 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. If CNTSC=0, CNTSCR* are permanently driven to `0x0100_0000`.

Note

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

A.4 System Timer overview

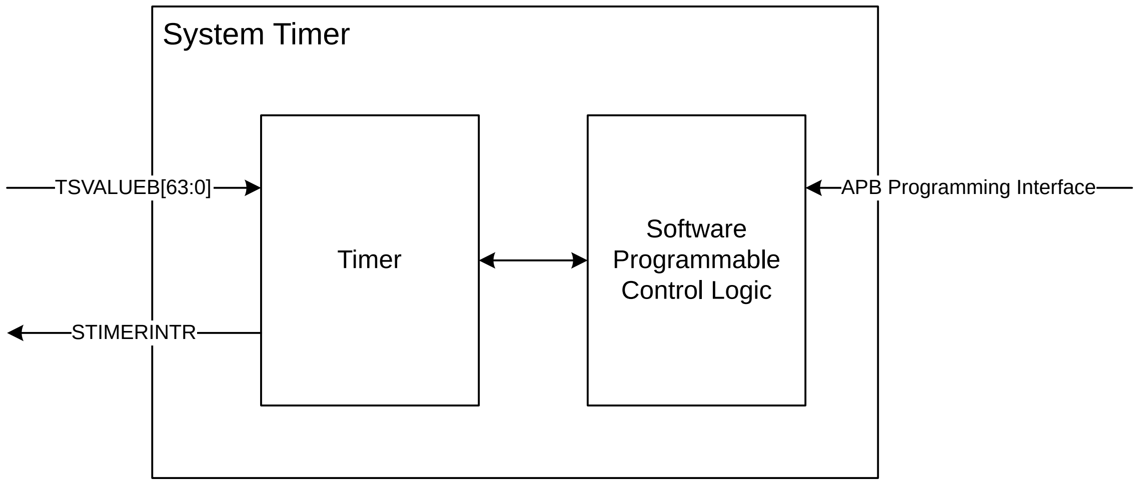
This section provides an overview of the System Timer.

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 A-1: CRSAS Ma2 System Timer block diagram


System Timer
TSVALUEB[63:0]
STIMERINTR
Timer
Software Programmable Control Logic
APB Programming Interface

A.4.1 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]
```

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:

`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.

A.5 System timer 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 defined by the system.

See Timestamp-based timers.

A.5.1 CNTBase Registers Summary

This section provides a summary of the CNTBase Registers.

The following table shows a summary of the CNTBase Registers.

Table A-9: CNTBase Frame Registers summary

Offset	Name	Type	Reset value	Description
`0x000`	CNTPCT[31:0]	RO	`0xXXXX_XXXX`	Physical Count Register. See CNTPCT, Counter-timer Physical Count register
`0x004`	CNTPCT[63:32]	RO	`0xXXXX_XXXX`	-

Offset	Name	Type	Reset value	Description
0x008 -0x00C	-	RAZ/WI	0x0000_0000	Reserved
0x010	CNTFRQ	RW	0x0000_0000	Counter Frequency register. See CNTFRQ, Counter-timer Frequency register
0x014 -0x01C	-	RAZ/WI	0x0000_0000	Reserved
0x020	CNTP_CVAL[31:0]	RW	0x0000_0000	Timer CompareValue register. See CNTP_CVAL, Counter-timer Physical Timer CompareValue register
0x024	CNTP_CVAL[63:32]	RW	0x0000_0000	-
0x028	CNTP_TVAL	RW	0xXXXX_XXXX	TimerValue register. See CNTP_TVAL, Counter-timer Physical Timer TimerValue register
0x02C	CNTP_CTL	RW	0x0000_000X	Timer Control register. See CNTP_CTL, Counter-timer Physical Timer Control register
0x030 -0x03C	-	RAZ/WI	0x0000_0000	Reserved
0x040	CNTP_AIVAL[31:0]	RO	0xXXXX_XXXX	AutoIncrValue register. See CNTP_AIVAL, AutoIncrValue register
0x044	CNTP_AIVAL[63:32]	RO	0xXXXX_XXXX	-
0x048	CNTP_AIVAL_RELOAD	RW	0xXXXX_XXXX	AutoIncrValue Reload register. See CNTP_AIVAL_RELOAD, AutoIncrValue Reload register
0x04C	CNTP_AIVAL_CTL	RW	0xXXXX_XXXX	AutoIncrValue Control register. See CNTP_AIVAL_CTL, AutoIncrValue Control register
0x050	CNTP_CFG	RO	0x0000_0001	Timer Configuration register. See CNTP_CFG, Configuration register
0x054 -0xFCC	-	RAZ/WI	0x0000_0000	Reserved
0xFD0	CNTP_PID4	RO	0x0000_0004	Peripheral ID4.
0xFD4 -0xFDC	-	RO	0x0000_0000	Reserved
0xFE0	CNTP_PID0	RO	0x0000_00B7	Peripheral ID0.
0xFE4	CNTP_PID1	RO	0x0000_00B0	Peripheral ID1.
0xFE8	CNTP_PID2	RO	0x0000_000B	Peripheral ID2.
0xFEC	CNTP_PID3	RO	0x0000_0000	Peripheral ID3.
0xFF0	CNTP_CID0	RO	0x0000_000D	Component ID0.
0xFF4	CNTP_CID1	RO	0x0000_00F0	Component ID1.
0xFF8	CNTP_CID2	RO	0x0000_0005	Component ID2.
0xFFC	CNTP_CID3	RO	0x0000_00B1	Component ID3.

A.5.2 Register descriptions

This section describes each of the System Timer registers.

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.

A.5.3 CNTPCT, Counter-timer Physical Count register

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

The following table shows the bit assignments.

Table A-10: CNTPCT register bit assignments

Bits	Name	Type	Reset value	Function
63:0	CountValue	RO	`0xXXXX_XXXX_XXXX_XXXX`	Physical count value.

A.5.4 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 A-11: CNTFRQ register bit assignments

Bits	Name	Typw	Reset value	Function
31:0	ClockFrequency	RW	`0xXXXX_XXXX_XXXX_XXXX`	Clock frequency.

A.5.5 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 A-12: CNTP_CVAL register bit assignments

Bits	Name	Type	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 counter timer. When the timer condition is met: • NTP_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.

A.5.6 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 A-13: CNTP_TVAL register bit assignments

Bits	Name	Type	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.

A.5.7 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 A-14: CNTP_CTL register bit assignments

Bits	Name	Type	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**.

Bits	Name	Type	Reset value	Function
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.
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.

A.5.8 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 A-15: CNTP_AIVAL register bit assignments

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

A.5.9 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 A-16: CNTP_AIVAL_RELOAD register bit assignments

Bits	Name	Type	Width	Reset value	Description
31:0	AutoIncrValue timer value	RO	32	`0x0000_0000_0000_0000`	Timer AutoIncrValue Reload.

A.5.10 CNTP_AIVAL_CTL, AutoIncrValue Control register

Control register for the Automatic Increment timer view.

The following table shows the bit assignments.

Table A-17: CNTP_AIVAL_CTL register bit assignments

Bits	Name	Type	Width	Reset value	Description
31:2	-	-	30	-	Reserved
1	CLR	WO	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	WO	1	`0xX`	Enables the AutoIncrValue register. Permitted values are: • 0: AutoIncrValue disabled. • 1: AutoIncrValue enabled.

A.5.11 CNTP_CFG, Configuration register

Provides timer configuration information.

The following table shows the bit assignments.

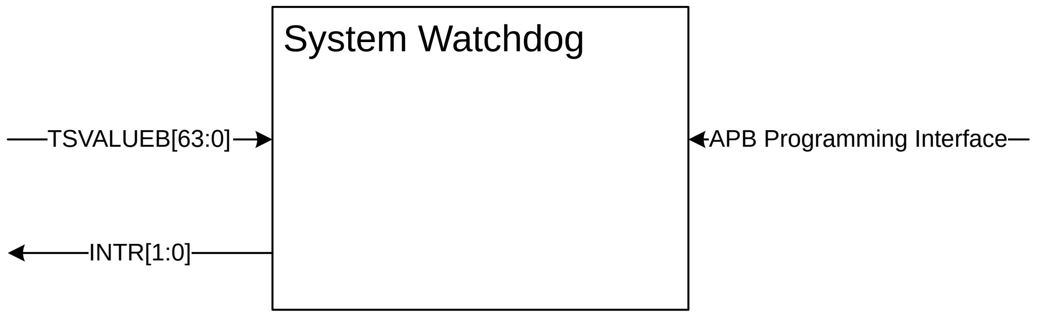
Table A-18: CNTP_CFG register bit assignments

Bits	Name	Type	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.

A.6 System Watchdog overview

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

Figure A-2: Block diagram of the System Watchdog



A.6.1 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 begins.

Second watch period behaviour

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.

A.6.2 System watchdog 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.

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

Table A-19: 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

A.6.3 Control Frame registers summary

This section provides a summary of the Control Frame registers.

The following table shows the Control Frame registers summary.

Table A-20: Control Frame registers summary

Offset	Name	Type	Reset value	Description
`0x000`	WCS	RW	`0x0000_0000`	WCS, Watchdog Control and Status register
`0x004`	-	RAZ/WI	`0x0000_0000`	Reserved
`0x008`	WOR	RW	`0x0000_0000`	WOR, Watchdog Offset register
`0x00c`	-	RAZ/WI	`0x0000_0000`	Reserved
`0x010`	WCV[31:0]	RW	`0x0000_0000`	WCV, Watchdog Compare Value register
`0x014`	WCV[63:32]	RW	`0x0000_0000`	-
`0x018-0xFC8`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFCC`	W_IIDR	RO	`0x0000_143B`	W_IIDR, Watchdog Interface Identification register
`0xFD0`	PIDR4	Read-only	`0x0000_0004`	Peripheral ID 4
`0xFD4 – 0xFDC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFE0`	PIDR0	Read-only	`0x0000_00B1`	Peripheral ID 0
`0xFE4`	PIDR1	Read-only	`0x0000_00B0`	Peripheral ID 1
`0xFE8`	PIDR2	Read-only	`0x0000_002B`	Peripheral ID 2
`0xFEC`	PIDR3	Read-only	`0x0000_0000`	Peripheral ID 3
`0xFF0`	CIDR0	Read-only	`0x0000_000D`	Component ID 0
`0xFF4`	CIDR1	Read-only	`0x0000_00F0`	Component ID 1
`0xFF8`	CIDR2	Read-only	`0x0000_0005`	Component ID 2
`0xFFC`	CIDR3	Read-only	`0x0000_00B1`	Component ID 3

A.6.4 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 A-21: Refresh Frame registers summary

Offset	Name	Type	Reset value	Description
`0x000`	WRR	RW	`0x0000_0000`	WRR, Watchdog Refresh register
`0x004-0xFC8`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFCC`	W_IIDR	RO	`0x0000_143B`	W_IIDR, Watchdog Interface Identification register
`0xFD0`	PIDR4	Read-only	`0x0000_0004`	Peripheral ID 4
`0xFD4 – 0xFDC`	-	RAZ/WI	`0x0000_0000`	Reserved
`0xFE0`	PIDR0	Read-only	`0x0000_00B0`	Peripheral ID 0
`0xFE4`	PIDR1	Read-only	`0x0000_00B0`	Peripheral ID 1
`0xFE8`	PIDR2	Read-only	`0x0000_002B`	Peripheral ID 2
`0xFEC`	PIDR3	Read-only	`0x0000_0000`	Peripheral ID 3
`0xFF0`	CIDR0	Read-only	`0x0000_000D`	Component ID 0
`0xFF4`	CIDR1	Read-only	`0x0000_00F0`	Component ID 1
`0xFF8`	CIDR2	Read-only	`0x0000_0005`	Component ID 2
`0xFFC`	CIDR3	Read-only	`0x0000_00B1`	Component ID 3

A.6.5 Register descriptions

This section describes each System Watchdog register.

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

A.6.6 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 A-22: WCS register bit assignments

Bits	Name	Type	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].

Bits	Name	Type	Reset value	Function
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.

A.6.7 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 A-23: WOR register bit assignments

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

A.6.8 WCV, Watchdog Compare Value register

The WCV register holds the compare value of the watchdog.

The following table shows the bit assignments.

Table A-24: WCV register bit assignments

Bits	Name	Type	Reset value	Function
63:0	WCV	RW	`0x0000_0000`	HHolds the current 64-bit compare value.

A.6.9 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 A-25: W_IIDR register bit assignments

Bits	Name	Type	Reset value	Function
31:24	ID	RO	`0x0`	Product identifier. `0x00` = System Watchdog.
23:20	-	RO	`0x0`	Reserved
19:16	ARCH	RO	`0x0`	Architecture version, v0.

Bits	Name	Type	Reset value	Function
15:12	REV	RO	`0x1`	Revision number for the component. `0x1` = r0p1
11:0	JEPCODE	RO	`0x43B`	Arm JEP106 code.

A.6.10 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 A-26: WRR register bit assignments

Bits	Name	Type	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.

Appendix B Revisions

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

Table B-1: Issue 0000-01

Change	Location
Initial issue of the document	-

Appendix C Technical changes between CRSAS Ma1 and Ma2

List of technical changes made between the CRSAS Ma1 and Ma2.

The following table lists the changes between the documents with links to the relevant topics.

Table C-1: Changes between the CRSAS Ma1 and the CRSAS Ma2

Change	Location
Ethos-U65 support added.	• Overview • Topology • Configuration options • NPU security mapping • NPU • NPUSPPORPL • Advance BR_NPU<m> power modes • Basic BR_NPU<m> power modes • Intermediate BR_NPU<m> power modes • Non-secure Access Configuration Register block • NPU<m> registers • NPUNSPORPL • NPUSPPORSL • Secure Access Configuration Register block • SYS_CONFIG2
Boot ROM support added.	New functionality added: • Configuration options • Read-only memory • ROM • System memory map overview • SECPPCINTCLR • SECPPCINTEN • SECPPCINTSTAT No functional change: • Peripheral Protection Controllers • NPU<m> registers • PERIPHNSPPC0 • PERIPHNSPPC1 • Timestamp-based timers registers • Timestamp-based watchdogs registers

Change	Location
Cryptocell 312 replaced with LCM, KMU, SAM. Added boot and secure provisioning example flows.	• Overview • Topology • Trusted Base System Architecture for Armv8-M • Configuration options • Clock control Q-Channel device interfaces • Input and output clocks • Functional reset inputs and outputs • Power Control Q-Channel and P-Channel Expansion Device interfaces • Power Domain ON Status Signals • Clocking infrastructure • Lifecycle Manager Non-Volatile Memory interface • Lifecycle Manager • PERIPHSPPPC1 • Peripheral Protection Controllers • LCM Debug Control Unit • Secure Debug Configuration Registers • Debug Authentication interface • Lifecycle Manager Lifecycle Indication interface • Lifecycle Manager Expansion interfaces • Lifecycle Manager Debug Control Unit interface • Lifecycle Manager LFSR seed interface • Sensor Alarm Interface • System boot flow • LCM_DCU_FORCE_DISABLE • Reset infrastructure • Secure Asset Provisioning Flow • Key Management Unit • Security Alarm Manager • Security Alarm Manager incoming events allocation • Security Alarm Manager Response Action Settings • Miscellaneous signals
Software syndrome SWSYN field added to the RESET_SYNDROME and SWRESET registers.	• RESET_SYNDROME • SWRESET
Addition of Q or P-Channel for AON power domain to support Warm reset entry, which replaces EXPWARMRESETREQ and EXPWARMRESETACK handshake.	• Functional reset inputs and outputs • Reset infrastructure • Power Control Q-Channel and P-Channel Expansion Device interfaces
Addition of WARMRESETREQ input and WARMRESETACK output, and addition of new field in RESET_SYNDROME to support this request.	• Functional reset inputs and outputs • Reset infrastructure • RESET_SYNDROME

Change	Location
RESET_SYNDROME .SWRESETREQ moved from bit 9 to bit 5.	• RESET_SYNDROME
New configuration, CPUWAITRST_MODE, to select CPU<n>WAIT to be reset using nWARMRESETAON or nCOLDRESETAON.	• Configuration options • CPUWAIT
New configuration, INITSVTORRST_MODE, to select whether Cold reset or Warm rest is used to reset the INITSVTOR<n> registers.	• Configuration options • INITSVTOR<n>
New software Warm reset request, SWWARMRESETREQ, field in SWRESET register, to allow software to request for a system Warm reset. Also added corresponding SWWARMRESETREQ field in RESET_SYNDROME register. Renamed SWRESETREQ to SWCOLDRESETREQ, and SWRSTREQSTATUS to SWCOLDRSTREQSTATUS	• Reset infrastructure • Miscellaneous signals • SWRESET • RESET_SYNDROME • System boot flow • Configuration options • Power-on and Cold reset handling • Secure Asset Provisioning Flow • Warm Reset generation
Clarification that EPU retention requires the core to enter sleep mode if: • CPDLPSTATE.ELPSTATE = OFF, (`0b11`) and • CPPWR.SU10 = `0b0` The other case does not require sleep mode: • CPDLPSTATE.ELPSTATE = RET, (`0b10`)	• Controlling PD_CPU<n>EPU power states • Controlling PD_CPU<n>EPU power states • Controlling PD_CPU0EPU power states
Clarification that DMA can only support up to 16 Channels.	• DMA
Update to support more CTI triggers beyond 8 bits.	• Cross Trigger Interface • Cross Trigger
Added Optional SDC-600 support	• Peripheral region • PERIPHNSPPC0 • PERIPHSPPPC0 • Secure Debug Channel registers • Overview • Debug Access • Interrupts

Change	Location
Change from Coresight SoC-600 to CoreSight SoC-600M.	• Cross Trigger Channel interface • Cross Trigger Interface • Cross Trigger, Basic debug configuration • Basic debug configuration • Main system configuration options • Debug Access • Debug infrastructure, • Full debug configuration • Debug System Access Region • CoreSight SoC-600M based Debug System implemented • CoreSight SoC-600M based Debug System not implemented • SYS_CONFIG0
Move to Type `0x9` Debug ROMs with CoreSight SoC-600M.	• SDSROM, Shared Debug System ROM • DSROM, Debug System ROM • CPU<n>ROM, CPU<n> Debug System ROM