Arm® Architecture Reference Manual Supplement, The Realm Management Extension (RME), for Armv9-A

Document number	ARM DDI 0615
Document quality	EAC
Document version	A.b
Document confidentiality	Non-confidential

Arm® Architecture Reference Manual Supplement, The Realm Management Extension (RME), for Armv9-A

Release information

Date	Version	Changes
2021/Nov/02	A.b	• Updated EAC release.
2021/Jun/23	A.a	• First EAC publication.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349 version 21.0

Product Status

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

The information in this Manual is at EAC quality, which means that all features of the specification are described in the manual.

Contents

Arm® Architecture Reference Manual Supplement, The Realm Management Extension (RME), for Armv9-A

Contents

Chapter A3 List of instructions

Glossary

Preface

About this book

This book is the Arm® Architecture Reference Manual Supplement, The Realm Management Extension (RME), for Armv9-A. This book describes the changes and additions to the Armv9-A architecture that are introduced by RME, and therefore must be read in conjunction with the *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile*.

It is assumed that the reader is familiar with the Armv8-A and Armv9-A architecture.

Conventions

Typographical conventions

The typographical conventions are:

italic

Introduces special terminology, and denotes citations.

bold

Denotes signal names, and is used for terms in descriptive lists, where appropriate.

`monospace`

Used for assembler syntax descriptions, pseudocode, and source code examples.

Also used in the main text for instruction mnemonics and for references to other items appearing in assembler syntax descriptions, pseudocode, and source code examples.

SMALL CAPITALS

Used for some common terms, for example IMPLEMENTATION DEFINED.

Used for a few terms that have specific technical meanings, and are included in the Glossary.

Red text

Indicates an open issue.

Blue text

Indicates a link. This can be:

- A cross-reference to another location within the document
- A URL, for example http://developer.arm.com

Numbers

Numbers are normally written in decimal. Binary numbers are preceded by `0b`, and hexadecimal numbers by `0x`. In both cases, the prefix and the associated value are written in a monospace font, for example `0xFFFF0000`. To improve readability, long numbers can be written with an underscore separator between every four characters, for example `0xFFFF_0000_0000_0000`. Ignore any underscores when interpreting the value of a number.

Pseudocode descriptions

This book uses a form of pseudocode to provide precise descriptions of the specified functionality. This pseudocode is written in a monospace font. The pseudocode language is described in the Arm Architecture Reference Manual.

Assembler syntax descriptions

This book contains numerous syntax descriptions for assembler instructions and for components of assembler instructions. These are shown in a `monospace` font.

Rules-based writing

This specification consists of a set of individual *content items*. A content item is classified as one of the following:

- Declaration
- Rule
- Goal
- Information
- Rationale
- Implementation note
- Software usage

Declarations and Rules are normative statements. An implementation that is compliant with this specification must conform to all Declarations and Rules in this specification that apply to that implementation.

Declarations and Rules must not be read in isolation. Where a particular feature is specified by multiple Declarations and Rules, these are generally grouped into sections and subsections that provide context. Where appropriate, these sections begin with a short introduction.

Arm strongly recommends that implementers read *all* chapters and sections of this document to ensure that an implementation is compliant.

Content items other than Declarations and Rules are informative statements. These are provided as an aid to understanding this specification.

Content item identifiers

A content item may have an associated identifier which is unique among content items in this specification.

After this specification reaches beta status, a given content item has the same identifier across subsequent versions of the specification.

Content item rendering

In this document, a content item is rendered with a token of the following format in the left margin: L_{iiiii}

- *L* is a label that indicates the content class of the content item.
- *iiiii* is the identifier of the content item.

Content item classes

Declaration

A Declaration is a statement that does one or more of the following:

- Introduces a concept
- Introduces a term
- Describes the structure of data
- Describes the encoding of data

A Declaration does not describe behavior.

A Declaration is rendered with the label *D*.

Rule

A Rule is a statement that describes the behavior of a compliant implementation.

A Rule explains what happens in a particular situation.

A Rule does not define concepts or terminology.

A Rule is rendered with the label *R*.

Goal

A Goal is a statement about the purpose of a set of rules.

A Goal explains why a particular feature has been included in the specification.

A Goal is comparable to a “business requirement” or an “emergent property.”

A Goal is intended to be upheld by the logical conjunction of a set of rules.

A Goal is rendered with the label *G*.

Information

An Information statement provides information and guidance as an aid to understanding the specification.

An Information statement is rendered with the label *I*.

Rationale

A Rationale statement explains why the specification was specified in the way it was.

A Rationale statement is rendered with the label *X*.

Implementation note

An Implementation note provides guidance on implementation of the specification.

An Implementation note is rendered with the label *U*.

Software usage

A Software usage statement provides guidance on how software can make use of the features defined by the specification.

A Software usage statement is rendered with the label *S*.

Additional reading

This section lists publications by Arm and by third parties.

See Arm Developer (http://developer.arm.com) for access to Arm documentation.

[1] *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile*. (ARM DDI 0608) Arm Ltd.

[2] *Arm® Architecture Reference Manual for Armv8-A architecture profile*. (ARM DDI 0487) Arm Ltd.

[3] *Arm® System Memory Management Unit Architecture supplement - The Realm Management Extension (RME), for SMMUv3*. (ARM IHI 0094) Arm Ltd.

[4] *Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture*. (ARM DDI 0598) Arm Ltd.

[5] *Arm® Architecture Reference Manual Supplement, Reliability, Availability, and Serviceability (RAS), for Armv8-A*. (ARM DDI 0587) Arm Ltd.

[6] *Arm® Architecture Reference Manual Supplement, The Scalable Vector Extension*. (ARM DDI 0584) Arm Ltd.

[7] *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*. (ARM IHI 0069) Arm Ltd.

Feedback

Arm welcomes feedback on its documentation.

Feedback on this book

If you have comments on the content of this book, send an email to errata@arm.com. Give:

- The title (Arm® Architecture Reference Manual Supplement, The Realm Management Extension (RME), for Armv9-A).
- The number (ARM DDI 0615 A.b).
- The page numbers to which your comments apply.
- The rule identifiers to which your comments apply, if applicable.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests PDFs only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the appearance or behavior of any document when viewed with any other PDF reader.

Progressive terminology commitment

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

We believe that this document contains no offensive terms. If you find offensive terms in this document, please contact terms@arm.com.

Chapter 1
Introduction

The Realm Management Extension (RME) is an extension to the Armv9 A-profile architecture. RME adds the following features:

- Two additional Security states, Root and Realm.
- Two additional physical address spaces, Root and Realm.
- The ability to dynamically transition memory granules between physical address spaces.
- Granule Protection Check mechanism.

RME is one component of the Arm Confidential Compute Architecture (Arm CCA). Together with the other components of the Arm CCA, RME enables support for dynamic, attestable, and trusted execution environments (*Realms*) to be run on an Arm PE.

R_{XKGPG} RME provides hardware-based isolation that allows execution contexts to run in different Security states and share resources in the system while ensuring that:

- Execution in the Realm Security state cannot be observed or modified by an agent associated with either the Non-secure Security state or the Secure Security state.
- Execution in the Secure Security state cannot be observed or modified by an agent associated with either the Non-secure Security state or the Realm Security state.
- Execution in the Root Security state cannot be observed or modified by an agent associated with any other Security state.
- Memory assigned to the Realm Security state cannot be read or modified by an agent associated with either the Non-secure Security state or the Secure Security state.
- Memory assigned to the Secure Security state cannot be read or modified by an agent associated with either the Non-secure Security state or the Realm Security state.
- Memory assigned to the Root Security state cannot be read or modified by an agent associated with any other Security state.

This specification uses the term *RME security guarantee* to describe the preceding properties.

R_{RBTNL} In this section, the term Memory is used to mean Locations and associated Allocation Tags.

I_{RLZQP} The RME architecture upholds the RME security guarantee.

I_{RYPWM} The RME architecture does not relax any security guarantees made by the Armv9-A architecture.

I_{ZLZWD} Software written for Non-secure state in EL1 or EL0 can run without modification in Realm Security state.

Chapter 2
Architecture Features and Extensions

RME inherits the rules for architectural features and extensions from Armv9-A [1]. This section describes changes to those rules, and defines any features added by RME.

2.1 Extensions and features defined by RME

R_{DQHBH} An architecture extension, the *Realm Management Extension* (RME), is introduced. RME is represented by the feature string FEAT_RME.

R_{PNMTT} A version for the Embedded Trace Extension, *Embedded Trace Extension version 1.2*, is introduced. Embedded Trace Extension version 1.2 is represented by the feature string FEAT_ETEv1p2.

I_{PFRMV} FEAT_ETEv1p2 covers changes to FEAT_ETEv1p1 to support FEAT_RME.

R_{HWDKJ} An architecture feature, the *RNG Trap*, is introduced. The RNG Trap feature is represented by the feature string FEAT_RNG_TRAP.

I_{VDRPL} FEAT_RNG_TRAP introduces an EL3 trap on the RNDR and RNDRRS registers. To allow implementations that only support software emulation of the RNDR and RNDRRS registers, FEAT_RNG_TRAP does not require FEAT_RNG.

2.2 Changes to existing features and extension requirements

R_{KZYFR} Any feature described as mandatory in Armv9-A [1] is mandatory for a PE that implements FEAT_RME, unless explicitly stated otherwise.

R_{WPLSL} Any feature described as prohibited in Armv9-A [1] is prohibited for a PE that implements FEAT_RME, unless explicitly stated otherwise.

R_{XGYHC} Any feature described as optional in Armv9-A [1] is optional for a PE that implements FEAT_RME, unless explicitly stated otherwise.

R_{JWYBV} A PE that implements FEAT_RME also implements:

- One or both of FEAT_RNG and FEAT_RNG_TRAP.

R_{PVBCD} If the Performance Monitors Extension is implemented, a PE that implements FEAT_RME also implements:

- FEAT_PMUv3p7.

R_{YNDXB} If the Memory Partitioning and Monitoring Extension is implemented, a PE that implements FEAT_RME also implements:

- FEAT_MPAMv1p1.

R_{RLCJV} Subject to export restrictions, a PE that implements FEAT_RME also implements the Armv9-A Cryptographic Extension. This gives the following features:

- FEAT_AES.
- FEAT_PMULL.
- FEAT_SHA1.
- FEAT_SHA256.
- FEAT_SHA3.
- FEAT_SHA512.

R_{QRJZN} Subject to export restrictions, a PE that implements FEAT_RME and the Scalable Vector Extension also implements:

- FEAT_SVE_AES.
- FEAT_SVE_PMULL.
- FEAT_SVE_SHA3.

I_{HPSXQ} If the Activity Monitors Extension is implemented, Arm strongly recommends the following features are implemented:

- FEAT_AMUv1p1.

R_{PKDXP} If the Trace Architecture is implemented, a PE that implements FEAT_RME also implements:

- FEAT_ETEv1p2.

I_{CSLWZ} Arm recommends that a PE that implements FEAT_RME also implements:

- FEAT_VMID16.
- FEAT_HAFDBS, with support for hardware update of both the Access flag and dirty state.

Chapter 3
AArch64 Exception model

This section details changes to the AArch64 Exception model.

RME introduces two additional Security states, Root and Realm. It also introduces a class of synchronous exception, Granule Protection Check exceptions. These two additions are covered in this chapter.

3.1 Exception levels

R_{KTTLB} A PE that implements FEAT_RME supports the following Exception levels:

- EL0.
- EL1.
- EL2.
- EL3.

I_{BPNKJ} Armv9-A [1] requires that if EL2 and EL3 are implemented, then FEAT_SEL2 is implemented. This means that a PE that implements FEAT_RME also implements Secure EL2.

3.2 Execution states

R_{XVVFH} A PE that implements FEAT_RME does not support AArch32 at EL3, EL2, and EL1.

I_{KYWVP} Support for AArch32 at EL0 is IMPLEMENTATION DEFINED.

3.3 Security states

R_{PTZDV} A PE that implements FEAT_RME has four Security states:

- Non-secure.
- Secure.
- Realm.
- Root.

I_{HKXZW} Secure and Non-secure states are inherited from Armv8-A.

R_{MSQZG} If the current Exception level is EL3, the PE is in Root state.

R_{WZFVW} If the current Exception level is not EL3, the Security state is defined by the value of SCR_EL3.{NSE, NS} as described in Table 3.1.

Table 3.1: SCR_EL3 and PE Security states

SCR_EL3.NSE	SCR_EL3.NS	Security state
0	0	Secure
0	1	Non-secure
1	0	Reserved
1	1	Realm

I_{YBGYV} The Security states are illustrated in Figure 3.1.

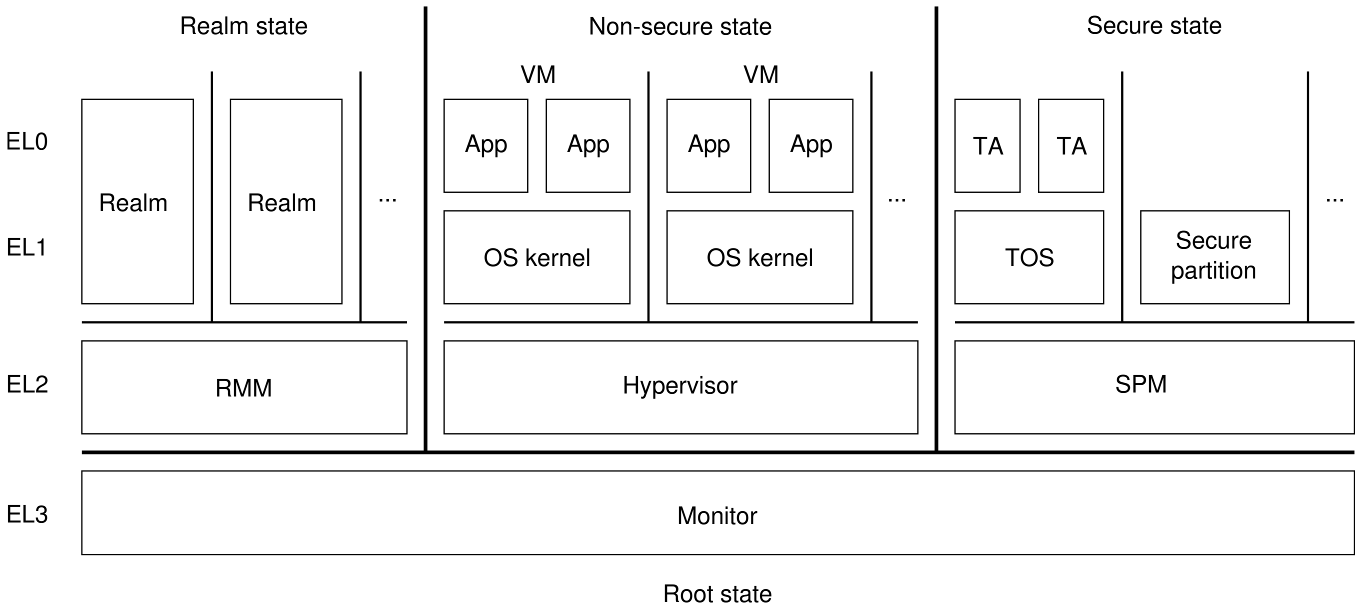


Figure 3.1: Security states

Where:

- RMM is Realm Management Monitor.
- SPM is Secure Partition Manager.

- VM is Virtual Machine.

R_{KXPGK} When performing an exception return from EL3 to a lower Exception level, if SCR_EL3.{NSE, NS} selects a reserved value, an illegal exception return results.

See also:

- 4.1 *Physical address spaces*

3.4 Exceptions

This subsection describes changes to exceptions, including the reporting of Granule Protection Check (GPC) faults. GPC faults are triggered by the GPC mechanism, which is described in *4.5 Granule Protection Checks*.

I_{PQFKF} Exceptions relating to GPC faults have the following properties:

- They do not compromise the availability of the system.
- They are synchronous.
- They are precise.
- They provide enough syndrome information for higher-privileged software to determine the lower-privileged software agent that experienced the fault.
- Higher-privileged software can make a Granule Protection Fault (GPF) visible to the lower-privileged software agent experiencing the fault.

3.4.1 Exceptions from GPC faults

I_{YRNGX} GPC fault is the collective term for the faults that can be returned by a granule protection check:

- GPF.
- Granule Protection Table (GPT) walk fault.
- GPT address size fault.
- Synchronous External abort on GPT fetch.

A GPC exception is a class of synchronous exception, which is used to report some GPC faults. Other GPC faults are reported as Instruction Abort or Data Abort exceptions.

How a GPC fault is reported as an exception depends on a number of factors:

- The type of access.
- The type of GPC fault.
- The Exception level that issued the access.
- The value of SCR_EL3.GPF.

R_{PYTGX} The following GPC faults are always reported as GPC exceptions:

- GPT address size fault.
- GPT walk fault.
- Synchronous External abort on GPT fetch.

I_{ZTKNY} GPC exceptions due to a synchronous External abort on GPT fetch are subject to SCR_EL3.EASE.

R_{BLYPM} A GPF at EL3 is reported synchronously as an Instruction Abort or Data Abort exception.

R_{VBZMW} If SCR_EL3.GPF == 1, a GPF at EL0, EL1, and EL2 is reported synchronously as a GPC exception.

R_{LXHQR} If SCR_EL3.GPF == 0, a GPF at EL0, EL1, or EL2 is reported synchronously as an Instruction Abort or Data Abort exception.

R_{YJLPJ} If GPCCR_EL3.GPCP == 0, all GPC faults are reported with a priority consistent with the GPC being performed on any access to physical address space. That is, for each existing synchronous External abort for an access as defined in the Armv8-A architecture, granule protection check faults are reported with immediately higher priority than the corresponding synchronous External abort for that access.

I_{ZQBDP} The priority order of synchronous aborts from a single stage of address translation is specified in section *AArch64 state prioritization of synchronous aborts from a single stage of address translation* in Armv8-A [2].

R_{LTMYZ} If GPCCR_EL3.GPCP == 1, then a GPC fault for the fetch of a Table descriptor for a stage 2 translation table walk might not be generated or reported. All other faults are reported with a priority consistent with the GPC being performed on any access to physical address space.

I_{YHXKR} The GPCCR_EL3.GPCP == 1 behavior is intended to permit hardware to elide granule protection checks on fetches of Table descriptors for stage 2 translations where it is safe to do so. This is in order to minimize the performance penalty of enabling granule protection checks. The analysis of whether the elision is acceptable to a security model includes a combination of the following factors:

- The use and style of memory encryption.
- The low probability of ciphertext being a valid translation table descriptor.
- The correct implementation of physical address space checks for read-sensitive locations (non-idempotent locations).

I_{RWGJH} If GPCCR_EL3.GPCP == 1, a decision to elide a granule protection check when fetching a translation table entry has to be re-evaluated when the entry content is processed:

- If the fetched entry is not a Table descriptor, then a granule protection check for the address of the fetched entry must be initiated and completed before the translation completes.
- Two granule protection checks can be initiated concurrently in such case, for the address of the fetched entry and for the content of the fetched entry, as long as the priority order for fault reporting is maintained.
- Arm strongly recommends that a granule protection check for the address of the fetched entry will also be performed in the case where the fetched entry generates a fault that would report syndrome information from that entry. Examples of such faults are Translation faults, Address Size faults, or External aborts. In this case, if the granule protection check results with a GPC fault, it is reported with priority as though GPCCR_EL3.GPCP == 0.

Regardless of the configuration of GPCCR_EL3.GPCP:

- It is implementation specific if a granule protection check for the address of a fetched entry is initiated concurrently with the fetch itself or only after the granule protection check is completed.
- The granule protection check must be completed before entry content is processed:
  - Where that is mandated by this document.
  - Where that is mandated by Armv8-A [2] rules for speculative operations.

3.4.2 Granule protection check exceptions

R_{XTYMC} GPC exceptions are synchronous, with ESR_EL3.EC == `0b01_1110`.

R_{FXMGJ} GPC exceptions are taken to EL3.

R_{BFJJV} On taking a GPC exception, the faulting physical address is saved in MFAR_EL3.

R_{QHPRZ} MFAR_EL3 is made UNKNOWN as a result of an exception return from EL3.

R_{CWWYV} On taking a GPC exception, the faulting Virtual Address is saved in FAR_EL3.

R_{DZDJV} On taking a GPC exception, if the fault was on an access as part of a stage 2 translation table walk, the faulting IPA is saved to HPFAR_EL2.

I_{WPWFG} HPFAR_EL2 is made UNKNOWN by an exception return from EL2, this is not the case for an exception return from EL3.

I_{XFQXD} The preferred exception return address for GPC exceptions is the address of the instruction that generates the exception. This follows the Armv8-A rules for synchronous exceptions which are not system calls.

I_FJFXQ The syndrome information provided on taking a GPC exception is illustrated here:

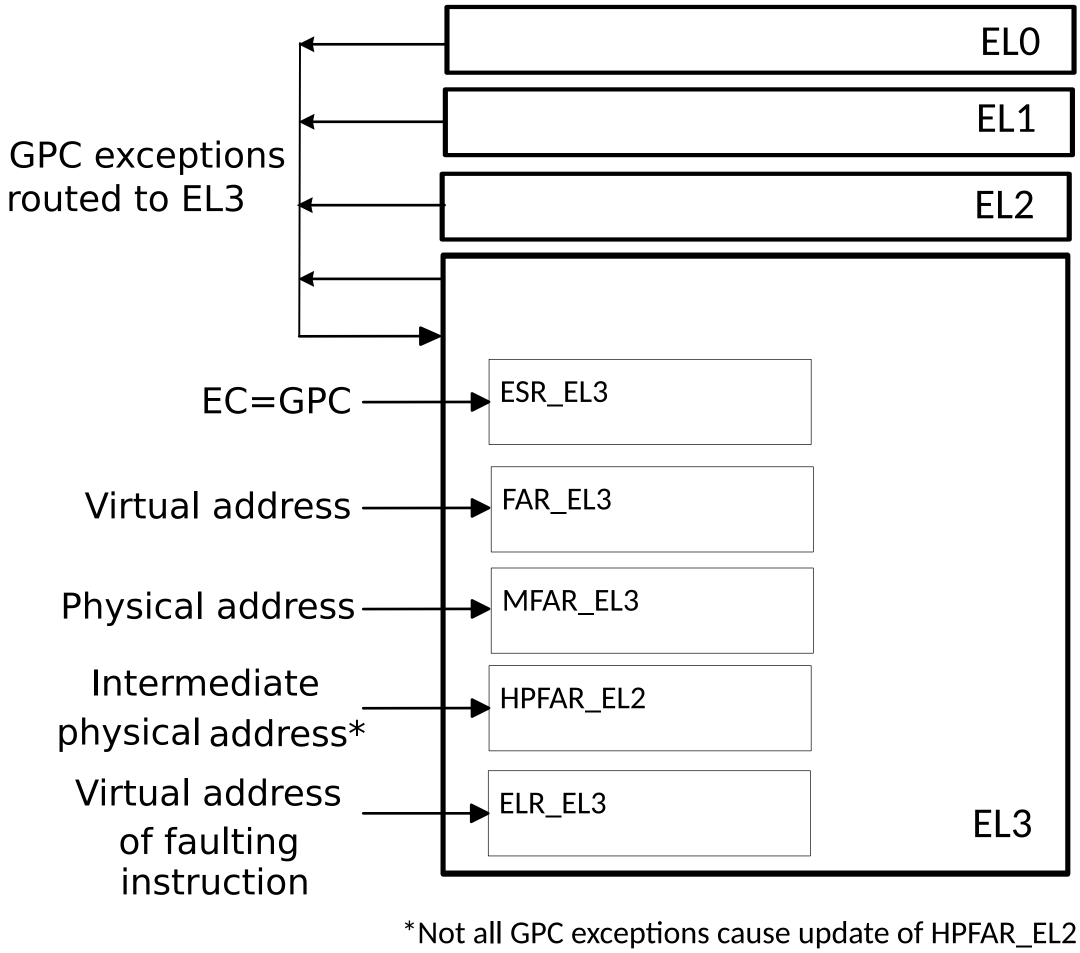


Figure 3.2: GPC syndrome

R_GMGRR The priority of reasons leading to a GPC fault is as follows:

Priority	Fault reported	Reason
1	GPT walk fault at Level 0	The configuration of GPCCR_EL3 is invalid
2	Granule protection fault at Level 0	A Secure, Realm or Root physical address exceeds GPCCR_EL3.PPS
3	GPT address size fault at Level 0	The base address in GPTBR_EL3.BADDR exceeds GPCCR_EL3.PPS
4	Synchronous External abort on GPT fetch at Level 0	An L0GPT fetch experiences an external abort
5	GPT walk fault at Level 0	An L0GPT entry is invalid
6	GPT address size fault at Level 0	An L0GPT entry contains an address exceeding GPCCR_EL3.PPS
7	Granule protection fault at Level 0	An L0GPT entry forbids access
8	Synchronous External abort on GPT fetch at Level 1	An L1GPT fetch experiences an external abort

Priority	Fault reported	Reason
9	GPT walk fault at Level 1	An L1GPT entry is invalid
10	Granule protection fault at Level 1	An L1GPT entry forbids access

I_{RFMWD} A GPC exception might occur at any point in the translation process that requires access to a physical address. For example, to perform a store at EL1, a PE would perform:

- Stage 1 translation for the accessed VA.
- Stage 2 translation for:
  - The IPA of each accessed stage 1 descriptor.
  - The output IPA from stage 1.
- Granule protection checks for:
  - The PA of each accessed stage 2 descriptor.
  - The PA of each accessed stage 1 descriptor.
  - The PA the accessed VA translated to.

R_{GVSNZ} If an instruction that stores to memory generates a GPC fault, the value of each memory location that instruction stores to is either:

- Unchanged if access to the location triggered the GPC fault.
- UNKNOWN for any location for which access did not trigger a fault or debug event.

I_{ZDKBX} Consistent with the general behavior of Data Aborts, when a load or store instruction results in accesses to two granules, the accesses to each granule are subject to granule protection checks.

See also:

- 4.5.2 *GPC faults*
- 15.1.5 *ESR_ELx*
- 15.1.14 *MFAR_EL3*

3.4.2.1 Delegating GPFs to lower Exception levels

I_{XWVCY} GPFs from EL0, EL1, or EL2 can be reported as Instruction Abort or Data Abort exceptions, or as GPC exceptions, controlled by SCR_EL3.GPF. The reported exception class determines the Exception level the exception is taken to. For GPFs taken from lower Exception levels, EL3 software might choose to delegate the exception to a lower Exception level. The syndrome information uses a similar format for GPC exceptions and Instruction and Data Aborts caused by a GPF to make it easy for EL3 software to emulate Instruction and Data Aborts as part of handling a GPC exception.

I_{NFKFH} An example of a GPC exception being delegated to EL2 is illustrated here:

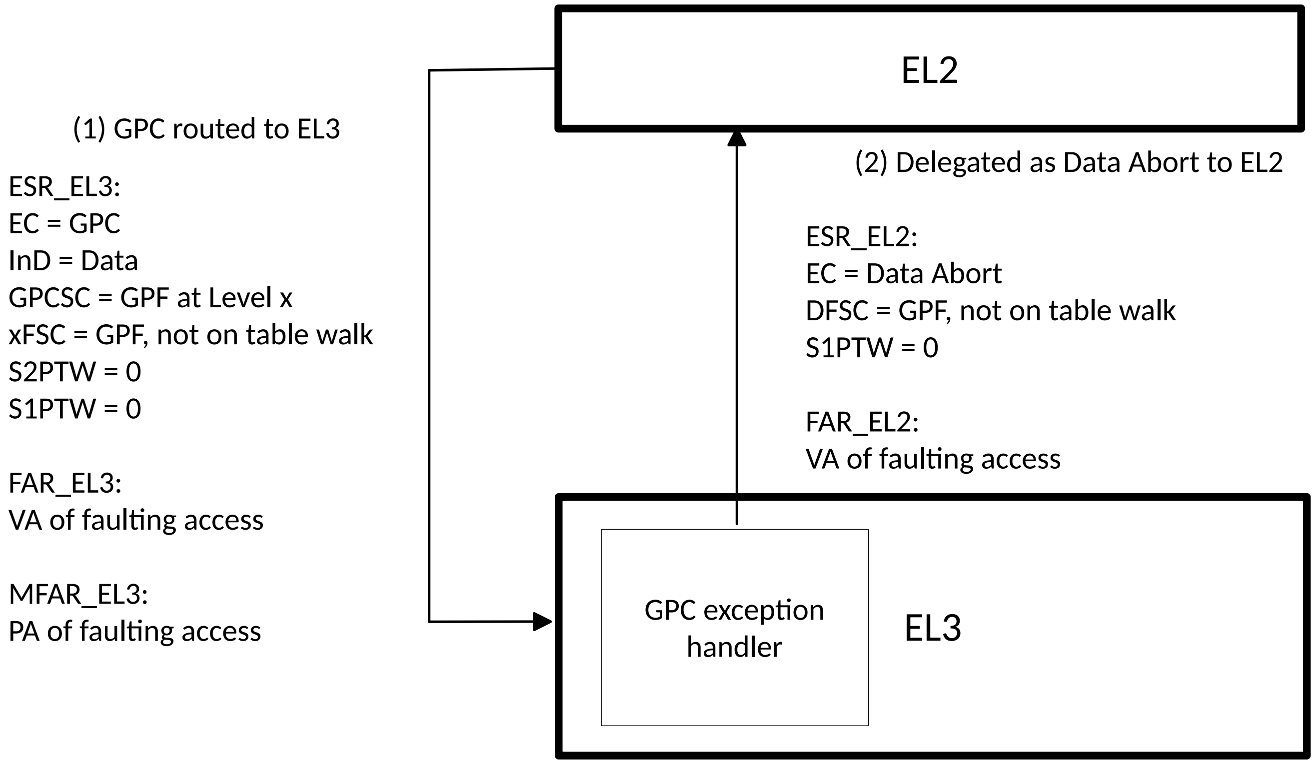


Figure 3.3: GPC delegation

See also:

- 4.5.2 *GPC faults*
- 15.1.5 *ESR_ELx*

3.4.3 Data and Instruction Abort exceptions

I_{DYCCK} Fault status codes are added to the Instruction Abort and Data Abort syndromes to report GPFs.

I_{NXSTP} Whether a Data Abort or Instruction Abort is signaled follows the standard rules for AArch64 described in Armv8-A [2].

R_{LZRHV} An Instruction Abort or Data Abort due to a GPF at EL3 is taken to EL3.

R_{NMCSJ} An Instruction Abort or Data Abort due to a GPF at EL2, or due to a GPF on an access for a stage 2 translation table, is taken to EL2.

I_{MHWXP} This includes GPFs on access for hardware update of stage 2 tables.

R_{WFTKR} An Instruction Abort or Data Abort due to a GPF at EL0 or EL1 is taken to:

- EL1 when HCR_EL2.{TGE, GPF} == {0,0}.
- EL2 when HCR_EL2.{TGE, GPF} != {0,0}.

R_{PRSHT} If an Instruction Abort or Data Abort is taken to EL2 due to a GPF on an access for a stage 2 translation table, the input IPA for the stage 2 translation is saved to HPFAR_EL2.

I_{CQGKM} On taking an Instruction Abort or Data Abort exception, the faulting virtual address is saved in the appropriate FAR_ELx. This is inherited from Armv8-A.

See also:

- 4.5.2 *GPC faults*
- 15.1.5 *ESR_ELx*
- 15.1.9 *HCR_EL2*

3.4.4 Asynchronous exception routing

R_{ZMVSM} Routing of asynchronous exceptions in Realm state is identical to that in Non-secure state.

I_{PVMFM} Table D1-8 (*Routing when both EL3 and EL2 are implemented*) in Armv8-A [2] describes the routing of asynchronous exceptions in AArch64. This table is unchanged by the introduction of FEAT_RME.

I_{YHNXM} Table D1-8 (*Routing when both EL3 and EL2 are implemented*) includes the SCR_EL3.NS bit.

To allow the representation of Realm state, FEAT_RME introduces SCR_EL3.NSE, which is treated as an extension of SCR_EL3.NS, with the following effects:

- Routing of asynchronous exceptions in Realm state and Non-secure state is identical and both have SCR_EL3.NS set to 1.
- Routing of asynchronous exceptions in Secure state, which has SCR_EL3.NS set to 0, is different due to the effects of SCR_EL3.EEL2.
- Routing of exceptions taken when the PE executes in EL3 is unaffected by SCR_EL3.{NS, NSE}.

Based on the effects, FEAT_RME makes no changes to Table D1-8.

Chapter 4
AArch64 Memory Model

This section details changes to the AArch64 memory model.

4.1 Physical address spaces

R_{YCZCD} A PE that implements FEAT_RME has four physical address spaces:

- Non-secure physical address space.
- Secure physical address space.
- Realm physical address space.
- Root physical address space.

I_{FBZJF} RME provides a mechanism to dynamically associate a memory physical Resource with one of the four physical address spaces, at a granularity which is one of the VMSA granule sizes.

4.2 Restrictions on the effects of speculation

I_{XDQQY} The term *speculative* has a specific meaning defined in the Armv8-A Glossary [2].

R_{WHDNB} The list of speculative operations is updated to include:

- Read accesses generated for a translation table walk for which the granule protection check for the address being accessed has not been architecturally resolved.

The existing rules around speculation in Armv8-A additionally apply to this speculative operation.

R_{QJCSL} When data is loaded under speculation with a GPC fault, it cannot be used to form an address, generate condition codes, or generate SVE predicate values to be used by other instructions in the speculative sequence, and the execution timing of any other instructions in the speculative sequence is not a function of the data loaded under speculation.

I_{DWDZB} This is similar to the rule forbidding speculation past a translation fault in Armv8-A [2].

R_{KLTMN} When stage 2 translation is enabled and a stage 1 translation table entry is loaded under speculation with a GPC fault, the Output address or Next-level table address from the entry cannot be used to form an address to be used by other fetches in the translation table walk.

R_{BTSYX} Granule protection checks apply to speculative execution. Any instruction fetched under speculation with a GPC fault cannot cause an update to any architectural or microarchitectural state as a result of speculative execution of the instruction, where the update of the state is dependent on the content of the instruction.

R_{CQSXX} If GPCCR_EL3.GPCP == 0, data from a translation table walk for which the granule protection check for the address being accessed has not been architecturally resolved can not be used to form an address for a subsequent read access or for generating syndrome information, until the granule protection check has passed.

I_{WRYMN} Permitting read accesses to locations for which the granule protection check has not been architecturally resolved means the GPT does not protect non-idempotent locations from these speculative read operations.

4.3 Limited ordering regions

I_{YPFYY} Limited ordering regions (LORegions) are defined in the Non-secure physical address space. LORegions cannot be defined in the Realm, Root, or Secure physical address spaces.

4.4 Caches

4.4.1 Point of Physical Aliasing

I_XDHLP The term *Location* defined in Armv8-A [2] means a byte that is associated with an address in a physical address space.

I_DXHTM For example, address `0x1000` in the Root physical address space is a different Location from address `0x1000` in the Secure physical address space.

R_QCQCS The term *Resource* means a physical entity that can be accessed at one or more Locations.

R_XGFKF A Resource *associated with* a physical address space is accessible in that physical address space.

I_YBCBT Examples of a Resource include:

- An MMIO register that is accessible at both the Location with address `0x2000` in the Non-secure physical address space, and at Location with address `0x2000` in the Secure address space.
- An SRAM that is accessible only at the Location with address `0x3000` in the Root physical address space.
- A byte of memory that can be accessible at a fixed address but in different physical address spaces, determined by a configuration option.

R_WSBQS The *Point of Physical Aliasing* (PoPA) is the point at which updates to one Location of a Resource are visible to all other Locations of that Resource, for accesses to that point of any memory type or cacheability attribute, for all agents that can access memory.

R_FVBBC The relationship between the PoPA and the *Point of Coherency* (PoC) is such that a clean of a written Location to the PoPA means that no agent in the system can subsequently reveal an old value of the Location by performing an invalidate operation to the PoC.

4.4.2 A64 cache maintenance instructions

R_FGTVF RME introduces new data cache maintenance operations.

System instruction	Instruction	Notes	Condition
`DC CIPAPA, Xt`	Clean and invalidate by physical address to PoPA	EL3 only	Present only if FEAT_RME is implemented.
`DC CIGDPAPA, Xt`	Clean and invalidate Allocation Tags and Data by physical address to PoPA	EL3 only	Present only if FEAT_RME and FEAT_MTE2 are implemented.

R_DLGCC `DC CIPAPA, Xt` and `DC CIGDPAPA, Xt` are system instructions with the following encodings:

Operation	op0	op1	CRn	CRm	op2
`DC CIPAPA, Xt`	0b01	0b110	0b0111	0b1110	0b001
`DC CIGDPAPA, Xt`	0b01	0b110	0b0111	0b1110	0b101

R_{FBXTC}

RFBXTC A `DC CIPAPA, Xt` or a `DC CIGDPAPA, Xt` instruction performs a clean and invalidate of data, and Allocation tags for `DC CIGDPAPA`, to the PoPA for all copies of the Location specified in the `Xt` argument to the instruction, for all caches in the Outer Shareable shareability domain.

RVRWVQ A `DC CIPAPA, Xt` or `DC CIGDPAPA, Xt` instruction is permitted to additionally affect other Locations of the Resource. If multiple Locations of the Resource have been written, it is CONSTRAINED UNPREDICTABLE which additional copies are cleaned to the PoPA. This CONSTRAINED UNPREDICTABLE behavior is guaranteed to be avoided if granule protection checks are configured to ensure that only one Location of the Resource is writable at any time.

RKFFFF `DC CIPAPA, Xt` and `DC CIGDPAPA, Xt` operations affect all caches before the PoPA, even if the caches are after the PoC and are otherwise invisible to the programmer.

RKRGQV `DC CIPAPA, Xt` and `DC CIGDPAPA, Xt` have the same ordering, observability, and completion behavior as VA-based cache maintenance instructions issued to the Outer Shareable shareability domain. This includes aspects relating to the minimum size of cachelines, indicated by CTR_EL0.DminLine.

RZKSNT In a system that contains caches associated with observers outside the Outer Shareable domain, then for each of those caches at least one of the following properties must apply:

- The cache is affected by DC PAPA operations. In this case, it is permitted for DC PAPA operations to be treated as invalidate, rather than clean and invalidate, operations for that cache.
- Any accesses from the cache that propagate into the Outer Shareable domain are subject to granule protection checks, and the system additionally provides one of the following properties:
  - The cache can only store Locations from the Non-secure physical address space.
  - Accesses from the cache are subject to translation controlled by the Security state associated with the cacheline.

RBCXGT The mechanism for granule protection checks for requesters that do not implement FEAT_RME is IMPLEMENTATION DEFINED but must allow Root firmware to configure a common GPT for all PE and non-PE requesters. Arm strongly recommends that the mechanism is as specified in SMMU for RME [3].

RLLCJF In the `DC CIPAPA, Xt` and `DC CIGDPAPA, Xt` instructions, the value of `Xt` is interpreted as:

Bits	Meaning
Xt[63]	NS
Xt[62]	NSE
Xt[61:52]	Reserved, RES0
Xt[51:0]	Physical address

The NS and NSE bits specify the target physical address space.

NSE	NS	Physical address space
0	0	Secure
0	1	Non-secure
1	0	Root
1	1	Realm

RYLDPH The `DC CIPAPA, Xt` and `DC CIGDPAPA, Xt` data cache maintenance instructions are UNDEFINED at EL2 and below.

RCZWDQ The `DC CIPAPA, Xt` and `DC CIGDPAPA, Xt` data cache maintenance instructions are not subject to granule protection checks.

4.4.3 Cache lockdown

I_{FNCXJ} Armv8-A [2] has IMPLEMENTATION DEFINED support for cache lockdown.

I_{GPGHT} The interaction of cache lockdown and existing data cache maintenance instructions is IMPLEMENTATION DEFINED, see Armv8-A [2] for more information.

R_{ZCSKB} Cache maintenance operations to the PoPA affect cache entries regardless of any lockdown status.

4.4.4 Cache maintenance by set/way and instruction Invalidate All operations

R_{YKDFZ} The effect of data and Allocation Tag cache maintenance operations by set/way, and the effect of Invalidate All instruction cache maintenance operations depend on the Security state when the operation is issued.

This behavior applies to the following operations:

- `DC ISW`, `DC CSW`, `DC CISW`
- `DC IGSW`, `DC CGSW`, `DC CIGSW`
- `DC IGDSW`, `DC CGDSW`, `DC CIGDSW`
- `IC IALLU`, `IC IALLUIS`

Security state issuing the operation	Entries required to be affected
Non-secure	Entries that are from the Non-secure PA space, and that match the other requirements of the operation.
Secure	Entries that are from the Non-secure or Secure PA space, and that match the other requirements of the operation.
Realm	Entries that are from the Non-secure or Realm PA space, and that match the other requirements of the operation.
Root	Entries that are from any PA space, and that match the other requirements of the operation.

4.5 Granule Protection Checks

Any access, after all enabled stages of translation, targets a physical address in one of the four physical address spaces.

This section introduces the mechanisms by which accesses to those physical address spaces are checked, including:

- Mechanism to determine the protection information for a particular physical address and physical address space.
- Allocation and invalidation behavior for TLB, data, and instruction caches.
- Configuration registers and descriptor formats for physical address space protection information.

4.5.1 GPC behavior overview

I_{WJCYJ} The granule protection mechanism permits association of a peripheral Resource with a physical address space to be performed by a Completer instead of the Requester, at a granularity finer than 4KB.

I_{PZSYC} The architecture permits caching of GPT information in a TLB, for implementations that chose to do so for area or performance reasons, in a manner sympathetic to existing TLB structures for VMSA in Armv8-A.

I_{BDSNC} The physical address space of an access is determined from the Security state of the Requester, as well as from stage 1 and stage 2 translation if enabled.

R_{BYRRZ} If granule protection checks are disabled (GPCCR_EL3.GPC == 0), accesses to all four address spaces are not subject to granule protection checks and cannot experience Granule Protection Check faults (GPC faults).

R_{GRGXY} If granule protection checks are enabled (GPCCR_EL3.GPC == 1), all accesses are subject to granule protection checks, except for fetches of GPT information and accesses governed by the GPCCR_EL3.GPCP control.

R_{XSWYP} If the Point of Coherency is before any level of cache and DC instructions to the PoC do not affect caches past the PoPA, it is IMPLEMENTATION DEFINED whether a data or unified cache maintenance by VA to the PoC instruction can generate a GPC fault.

I_{RLDTY} If granule protection checks are enabled (GPCCR_EL3.GPC == 1), an access might experience one of the following GPC faults:

- Granule Protection Fault.
- GPT walk fault.
- GPT address size fault.
- Synchronous External abort on GPT fetch.

R_{THJVJ} GPT walks are made to the Root physical address space and are not subject to granule protection checks.

I_{QHSZT} The GPCCR_EL3.GPCP control governs behavior of granule protection checks on fetches of stage 2 Table descriptors.

See also:

- 3.4.1 *Exceptions from GPC faults*
- 15.1.27 *GPCCR_EL3, Granule Protection Check Control Register*

4.5.2 GPC faults

R_{JWCSM} If the granule protection check for an access requires use of configuration in GPCCR_EL3, and the configuration of GPCCR_EL3 is invalid, the access fails as *GPT walk fault at level 0*.

The configuration of GPCCR_EL3 is invalid if any of the following are true:

- Any field is programmed to a reserved value.
- Any field is programmed to an invalid value, as specified in the definition of GPCCR_EL3.

R_{KYZMZ} If the granule protection check for an access requires consumption of any field in an invalid GPT entry, the access fails as *GPT walk fault at level x*, where x is the level of the invalid GPT entry.

R_{XVCKY} If the granule protection check for an access requires use of the configured base address in GPTBR_EL3.BADDR, and the base address exceeds the configured address size in GPCCR_EL3.PPS, the access fails as *GPT address size fault at level 0*.

R_{JCGMZ} If the granule protection check for an access requires consumption of a GPT Table descriptor with an address that exceeds the value configured in GPCCR_EL3.PPS, the access fails as *GPT address size fault at level 0*.

R_{DFCHJ} If a fetch of GPT information to check an access experiences an External abort, the access fails as *synchronous External abort on GPT fetch at level x*, where x is the level of the fetch that experienced the External abort.

R_{RLQVP} If a RAS error is detected on a fetch of GPT information to check an access, the access fails as *synchronous External abort on GPT fetch at level x* where x is the level of the fetch that consumed the RAS error.

R_{CPDSB} If a Non-secure physical address input to the granule protection check exceeds the physical address range specified by GPCCR_EL3.PPS, the access does not experience any GPC faults.

R_{JFFHB} If a Secure, Realm or Root physical address input to the granule protection check exceeds the physical address range specified by GPCCR_EL3.PPS, the access fails as *Granule protection fault at Level 0*.

R_{DQPWS} An access is not permitted by the GPT if it is made to a physical address space not permitted according to the GPI value returned by the GPT lookup.

R_{HDDNW} If an access is not permitted by the GPT, the access fails as *Granule protection fault at Level x*, where x is the level of the GPT entry that the access was checked against.

R_{VJLXG} Accesses are checked against the GPC configuration for the physical granule being accessed, regardless of the translation configuration for stage 1 and stage 2.

I_{KLTDM} For example, if GPCCR_EL3.PGS is configured to a smaller granule size than the translation granule size configured for stage 1 and stage 2 translation, accesses are checked at the GPCCR_EL3.PGS granule size.

See also:

- 3.4.2 *Granule protection check exceptions*

4.5.3 GPT caching and invalidation

R_{YJGPL} All fetches of GPT information use Normal memory types.

R_{RKFVK} The Cacheability and Shareability attributes of GPT fetches are configured in GPCCR_EL3.

I_{CDFPQ} Fetched GPT information might be cached in a data cache, according to the Normal memory Cacheability attributes and allocation hints configured in GPCCR_EL3.

I_{ZJYLQ} The Cacheability of GPT fetches is exclusively controlled by GPCCR_EL3.{IRGN, ORGN} and is not affected by the SCTLR_ELx.C or HCR_EL2.{CD, DC} control bits.

R_{XNFGN} GPT fetches are made with behavior consistent with PBHA being disabled or programmed to zero, regardless of the PBHA configuration at stage 1 and stage 2.

R_{YMSWK} GPT entries are permitted to be cached in TLBs combined with stage 1 and stage 2 information, as long as the requirements of TLB invalidation instructions are met.

R_{VFKSY} Consistent with TLB behavior at reset in Armv8-A [2], TLBs containing GPT information are disabled at reset. Any IMPLEMENTATION DEFINED or UNKNOWN GPT information in TLBs has no effect on accesses until granule protection checks, or any stages of translation are enabled.

R_{YMRVT} GPT information cached in a TLB is permitted to be shared across multiple PEs, except for PEs with GPCCR_EL3.GPC == 0 and all stages of translation disabled.

R_XLDKK For two PEs that are permitted to share GPT information cached in TLBs, if the configuration of GPCCR_EL3, GPTBR_EL3, and the GPT is not consistent across those PEs, the behavior on one PE is a CONSTRAINED UNPREDICTABLE choice of:

- The configuration for that PE.
- The configuration of the other PE.
- A combination of the configuration of the two PEs.

I_BSPQD To avoid CONSTRAINED UNPREDICTABLE behavior, Root firmware must ensure that both:

- Before GPCCR_EL3.GPC is set to 1, GPCCR_EL3 and GPTBR_EL3 are otherwise configured consistently with other PEs.
- Before enabling any stage of translation, GPCCR_EL3.GPC is set to 1.

R_RQCBQ A level 0 GPT entry is reachable if the entry is in the configured physical address range of GPTBR_EL3 and GPCCR_EL3.

R_MGSTK A level 1 GPT entry is reachable if a reachable and valid or previously cached level 0 GPT entry points to it.

R_BFQRM GPT entries may only be fetched if they are reachable.

R_QBKYP GPT entries may only be cached in a TLB if they are reachable and valid.

I_JMYRB Because GPT entries are permitted to be cached in a TLB if they are reachable and valid, translations that result in a Granule Protection Fault are permitted to be cached in a TLB.

R_PCYQZ TLB invalidation instructions for maintenance of GPT entries cached in a TLB are described with one of the following syntaxes:

- `TLBI RPA{L}OS, <Xt>.`
- `TLBI PAALLOS.`
- `TLBI PAALL.`

R_DRHKK The full set of TLB maintenance instructions that invalidate cached GPT entries is:

- `TLBI RPAOS, <Xt>.`
- `TLBI RPALOS, <Xt>.`
- `TLBI PAALLOS.`
- `TLBI PAALL.`

R_HLYDL The `TLBI *PA*` operations are system instructions with the following encodings:

System instruction	op0	op1	CRn	CRm	op2
TLBI RPAOS, <Xt>	0b01	0b110	0b1000	0b0100	0b011
TLBI RPALOS, <Xt>	0b01	0b110	0b1000	0b0100	0b111
TLBI PAALLOS	0b01	0b110	0b1000	0b0001	0b100
TLBI PAALL	0b01	0b110	0b1000	0b0111	0b100

R_YBBZK The `TLBI *PA*` instructions are only present at EL3. They are UNDEFINED at EL2 and below.

R_NBJFD `TLBI *PA*` instructions invalidate GPT information cached in TLB entries, including in intermediate TLB caching structures, according to the requirements specified in this section.

I_JJHVQ Armv8 permits a range of TLB implementation styles, including TLB caching structures that store entries that combine information from stage 1 and stage 2 translation table entries.

GPT information is permitted to be cached in combination with information from stage 1 and stage 2 translation table entries, as long as the requirements for invalidation of GPT information by `TLBI *PA*` operations are met. For example:

- An implementation that caches GPT information separately from stage 1 and stage 2 information is only required to invalidate GPT information as a result of a `TLBI *PA*` operation.
- An implementation that caches entries that combine stage 2 Output Address information with GPT information must invalidate all such entries in response to a `TLBI PAALLOS` operation.
- An implementation that caches entries that combine information from stage 2 level 2 Table descriptors with GPT information must invalidate those entries in response to a `TLBI *PA*` operation that matches the next-level address of those level 2 Table descriptors. It is not required to invalidate those entries on receipt of a `TLBI *PA*` that matches the physical address that the level 2 descriptor was fetched from.

R_{XZTJV} A `TLBI RPA*` instruction applies to TLB entries containing GPT information relating to the supplied physical address.

R_{ZDVNB} A `TLBI PAALL*` instruction applies to all TLB entries containing GPT information.

R_{BKJTM} A `TLBI PAALL*` instruction also applies to any TLB entry derived from GPC configuration register fields that are permitted to be cached in a TLB.

I_{JQRVK} The other syntax is the same as for Armv8-A. This means:

- `{R}` is a specifier denoting range-based invalidation.
- `{L}` is an optional specifier that reduces the scope of the invalidation to cached GPT entries fetched from the final level of the GPT walk.
- `{OS}` denotes that the TLBI applies to all the TLBs in the Outer Shareable domain. `TLBI *PA*` operations without `OS` are only required to apply for the PE executing the operation.
- `<Xt>` denotes that the instruction takes an X register as an argument to pass additional information about the invalidation scope.

R_{LRKLF} For `TLBI *PA*` instructions, Outer Shareable scope is sufficient to affect all TLBs in the system.

I_{SXTLQ} The `TLBI *PA*` operations do not have an nXS qualifier and always behave as though they are issued without an nXS qualifier.

I_{FRSHC} Armv8-A [2] has IMPLEMENTATION DEFINED support for TLB lockdown, and the interaction between TLB lockdown and existing data TLB maintenance instructions is IMPLEMENTATION DEFINED.

R_{BFXRL} `TLBI *PA*` operations affect TLB entries containing GPT information regardless of any TLB lockdown configuration.

R_{SGDDB} For `TLBI RPAOS, <Xt>` and `TLBI RPALOS, <Xt>` instructions, the value of `Xt` is interpreted as follows:

Bits	Meaning	Notes
[63:48]	Reserved	RES0
[47:44]	SIZE	See description of SIZE
[43:40]	Reserved	RES0
[39:0]	Address	See description of BaseADDR

R_{KTVYX} The encoding of BaseADDR depends on GPCCR_EL3.PGS as follows:

GPCCR_EL3.PGS	Size indicated	BaseADDR
`0b00`	4KB	BaseADDR[51:12] = `Xt`[39:0]
`0b10`	16KB	BaseADDR[51:14] = `Xt`[39:2]
`0b01`	64KB	BaseADDR[51:16] = `Xt`[39:4]

Other bits of BaseADDR are treated as zero, to give the Effective value of BaseADDR.

If GPCCR_EL3.PGS is configured to a reserved value, no TLB entries are required to be invalidated.

R_{RZYDS} The encoding of SIZE is:

Value	Meaning
0b0000	4KB
0b0001	16KB
0b0010	64KB
0b0011	2MB
0b0100	32MB
0b0101	512MB
0b0110	1GB
0b0111	16GB
0b1000	64GB
0b1001	512GB
Otherwise	Reserved

R_{NKBRB} A TLBI RPA*, <Xt> instruction performs range-based invalidation, and invalidates TLB entries starting from the address in BaseADDR, within the range as specified in the SIZE field.

If SIZE gives a range smaller than the configured physical granule size in GPCCR_EL3.PGS, then the Effective value of SIZE is taken to be the size configured by GPCCR_EL3.PGS.

If the Effective value of BaseADDR is not aligned to the size of the Effective value of SIZE, no TLB entries are required to be invalidated.

If SIZE is a reserved value, no TLB entries are required to be invalidated.

If GPCCR_EL3.PGS is configured to a reserved value, no TLB entries are required to be invalidated.

R_{TMCTS} The TLBI *PA* operations have the same rules around ordering, observability, and completion as all other TLBI instructions.

R_{PLYZN} The TLBI RPAOS instruction invalidates TLB entries containing GPT information from any level of the GPT walk relating to the supplied physical address.

R_{VLLLY} The TLBI RPALOS instruction invalidates TLB entries containing GPT information from the final level of the GPT walk relating to the supplied physical address.

I_{ZZJVG} Consistent with all other TLBI instructions, over-invalidation is permitted, and under-invalidation is not.

4.5.4 Table formats

R_{QKHMJ} The in-memory structure that describes the association of physical granules with physical address spaces is called the *Granule Protection Table* (GPT).

R_{JLXZV} A successful GPT lookup resolves an input physical address to the *Granule Protection Information* (GPI) for that address.

R_{TRVSY} A GPT descriptor is one of a Table, Block, Contiguous or Granules descriptor.

R_{JXNXP} A GPT descriptor is eight bytes.

R_{BQHPD} All structures in the GPT are little-endian.

I_{KFVNY} All GPT entries are naturally-aligned in memory.

R_{VXNGT} The GPT has two levels of lookup.

R_{TRCQY} All valid entries in a level 0 GPT are GPT Block or GPT Table descriptors.

R_{TXFXH} A level 0 GPT entry that is not a GPT Block or GPT Table descriptor is invalid.

R_{DCTFM} All valid entries in a level 1 GPT are GPT Contiguous or GPT Granules descriptors.

R_{TPBZN} A level 1 GPT entry that is not a GPT Contiguous or GPT Granules descriptor is invalid.

R_{XNKFZ} A GPT entry is invalid if any of the following are true:

- A field in the entry is configured with an encoding marked as reserved.
- A bit location in the entry marked as RES0 is nonzero.

I_{XJKRS} This is to increase the probability of detecting errors relating to a loss of integrity of the memory holding the GPT.

4.5.4.1 GPT Table descriptor

R_{RCTBJ} A GPT Table descriptor contains a pointer to the base address of a next-level table, and fields describing properties relating to the remaining levels of walk.

R_{HKPQF} The format of a GPT Table descriptor is described as follows:

Bits	Name
[63:52]	Reserved, RES0
[51:12]	Next-level Table Address
[11:4]	Reserved, RES0
[3:0]	`0b0011` Table descriptor

R_{DBTFW} The alignment of the Next-level Table Address depends on the value of GPCCR_EL3.PGS as follows:

Descriptor bits [s-p-2:12] are RES0, where:

- *s* is derived from GPCCR_EL3.L0GPTSZ as follows:

GPCCR_EL3.L0GPTSZ	Size indicated	*s*
`0b0000`	1GB	30
`0b0100`	16GB	34
`0b0110`	64GB	36
`0b1001`	512GB	39

- *p* is derived from GPCCR_EL3.PGS as follows:

GPCCR_EL3.PGS	Size indicated	*p*
`0b00`	4KB	12

GPCCR_EL3.PGS	Size indicated	*p*
`0b10`	16KB	14
`0b01`	64KB	16

I_{BFKHR} Level 1 tables are aligned to their size in memory. The size of level 1 tables is determined by GPCCR_EL3.PGS and GPCCR_EL3.L0GPTSZ.

4.5.4.2 GPT Block descriptor

R_{NKQNF} The format of a Block descriptor is described as follows:

Bits	Name
[63:8]	Reserved, RES0
[7:4]	GPI value
[3:0]	`0b0001` Block descriptor

R_{PLSSK} GPT information from a level 0 GPT Block descriptor is permitted to be cached in a TLB as though the block is a contiguous region of granules each of the size configured in GPCCR_EL3.PGS.

R_{YNKWN} A `TLBI RPA*` operation is only required to invalidate cached information from a level 0 GPT Block descriptor if the range encoded in the SIZE field of the invalidation covers the full address range of the Block, as advertised in GPCCR_EL3.L0GPTSZ.

R_{GXNNL} Granule protection checks continue to be made correctly, even if a TLBI is not issued, when GPT configuration is changed between the two following structures:

- A level 0 GPT Block descriptor indicating a GPI value for a region.
- A level 0 GPT Table descriptor pointing at a level 1 table of Contiguous or Granules descriptors that have the same GPI value as the level 0 Block descriptor.

In the scenario where a level 0 Table descriptor is replaced with a level 0 Block descriptor, the hardware may continue to access the level 1 Table until completion of a non-Last-level TLBI by PA, targetting at least the full address range of the level 0 descriptor. This means that the memory containing the level 1 Table cannot be reclaimed for other uses until completion of that TLBI by PA operation.

4.5.4.3 GPT Granules descriptor

R_{GQPWL} If bits [3:0] of a level 1 GPT entry are a valid GPI encoding, the entry is a GPT Granules descriptor.

R_{HJWQH} An 8-byte GPT Granules descriptor contains the GPI values for 16 physical granules.

R_{QDCZJ} The GPI values within one Granules descriptor are indexed as follows:

GPCCR_EL3.PGS	Size indicated	Within Granules descriptor
`0b00`	4KB	i = PA[15:12]
`0b10`	16KB	i = PA[17:14]
`0b01`	64KB	i = PA[19:16]

The GPI value to use is bits [(4*i) + 3 : (4*i)] of the descriptor.

R_GYQGW The encoding of a GPI field is:

Value	Meaning
0b0000	No accesses permitted
0b1000	Accesses permitted to Secure physical address space only
0b1001	Accesses permitted to Non-secure physical address space only
0b1010	Accesses permitted to Root physical address space only
0b1011	Accesses permitted to Realm physical address space only
0b1111	All accesses permitted
Otherwise	Reserved

I_YDVBY The GPI encoding "All accesses permitted" might be used for mapping peripherals that perform register banking based on the physical address space of an access.

4.5.4.4 GPT Contiguous descriptor

R_BSSVP If bits [3:0] of a level 1 GPT entry are 0b0001, the entry is a GPT Contiguous descriptor.

R_SPSCW The format of a GPT Contiguous descriptor is:

Bits	Description
[63:10]	Reserved, RES0
[9:8]	Contig
[7:4]	GPI
[3:0]	0b0001 (Contiguous descriptor)

R_BFCGF The encoding of the Contig field is:

Value	Meaning
0b00	Reserved
0b01	2MB
0b10	32MB
0b11	512MB

I_MVQFG There is no encoding for a 64KB contiguous region for the case where PGS is set to 4KB. If PGS is 4KB, it is permitted for an implementation to treat a GPT Granules descriptor containing 16 identical GPI values as a 64KB block region.

R_MNZWK Information from a GPT Contiguous descriptor is permitted to be cached in a TLB or walk cache for an input address range up to the size indicated by the Contig field.

RCZJSQ Contiguous regions are naturally-aligned.

IQJZQH For example, if the Contig field in the Contiguous descriptor for address `0x80004000` indicates a 2MB contiguous region, the region is `0x80000000` to `0x801FFFFF`.

RRQBNP GPT entries marked for contiguity are permitted but not required to be cached as block entries.

RSSKBB TLB Invalidation of GPT information is only guaranteed by TLB maintenance of the full range of the contiguity.

IKBTDW For example, this might be achieved by executing a `TLBI RPALOS, <Xt>` instruction covering the full range of the contiguous GPT region.

INZJDP This requirement on TLBI scope is intended to be the same as the behavior of the Contiguous bit in Armv8-A [2].

RSPLJH If any of the GPI values in GPT descriptors within the range specified by a Contig field differ from each other, then the GPT Contiguous descriptor has been *misprogrammed*.

RSMQTZ In the absence of other faulting conditions, if a GPT Contiguous descriptor has been misprogrammed, and for an access to a Location within the range specified by Contig, it is CONSTRAINED UNPREDICTABLE whether:

- The access succeeds as though its PA space is permitted by a programmed GPI value in the range.
- The access experiences a GPF consistent with the access not being permitted by one of the GPI values configured for the range.

RNNHCF In the absence of both misprogramming and faulting conditions, if a GPT Contiguous descriptor has Contig configured to one value, and other GPT Granules descriptors or Contiguous descriptors within the range indicated by that Contig field are all configured with the same GPI values, then accesses to that range are correctly checked against the GPI value programmed for the range.

IJMVJS This behavior is intended to be the same as the level 2 behavior that is specified in the FEAT_BBM feature, but with the option of TLB Conflict aborts removed.

See also:

- *4.5.2 GPC faults*
- *4.5.3 GPT caching and invalidation*

4.5.5 Lookup process

RNGQRV All accesses made by the MMU to the GPT are 64-bit single-copy atomic.

IGRXPD As an overview, the decoding of index information from a physical address input into the GPT lookup is as follows:

PA bits	Interpretation
PA[51:t]	Only applies if $t < 52$. Checked against GPCCR_EL3.PPS
PA[t-1:s]	Index into level 0 table
PA[s-1:p+4]	Index into level 1 table
PA[p+3:p]	Index of GPI within level 1 table entry

Where:

- The bit position t has the same value as the configured protected physical address size, decoded from GPCCR_EL3.PPS.
- The bit position s has the same value as the supported L0GPT entry size, decoded from GPCCR_EL3.L0GPTSZ.
- The bit position p has the same value as the address width of the physical granule size configured in

GPCCR_EL3.PGS:

- `0b00`, 4KB, $p = 12$
- `0b10`, 16KB, $p = 14$
- `0b01`, 64KB, $p = 16$

Tables at each level of the GPT are indexed by the input physical address bits, according to the values of GPCCR_EL3.{PPS, PGS, L0GPTSZ}.

R_{RDYKY} The level 0 table is indexed by PA bits as follows:

GPCCR_EL3.PPS	Level 0 index
0b000	PA[31:*s*]
0b001	PA[35:*s*]
0b010	PA[39:*s*]
0b011	PA[41:*s*]
0b100	PA[43:*s*]
0b101	PA[47:*s*]
0b110	PA[51:*s*]

The bit position *s* has the same value as the supported L0GPT entry size, decoded from GPCCR_EL3.L0GPTSZ.

If GPCCR_EL3.PPS is configured for a range smaller than or equal to the range advertised in GPCCR_EL3.L0GPTSZ, the level 0 table contains only one entry, at offset zero from the configured table base.

R_{RSHYW} The level 1 table is indexed by PA bits as follows:

GPCCR_EL3.PGS	Size indicated	Level 1 index
0b00	4KB	PA[*s*-1:16]
0b10	16KB	PA[*s*-1:18]
0b01	64KB	PA[*s*-1:20]

The bit position *s* has the same value as the supported L0GPT entry size, decoded from GPCCR_EL3.L0GPTSZ.

I_{LJQCV} The amount of memory occupied by a level 1 table depends on GPCCR_EL3.L0GPTSZ and GPCCR_EL3.PGS as follows:

L0GPTSZ	PGS=4KB	PGS=16KB	PGS=64KB
0b0000, 30 bits	128KB	32KB	8KB
0b0100, 34 bits	2MB	512KB	128KB
0b0110, 36 bits	8MB	2MB	512KB
0b1001, 39 bits	64MB	16MB	4MB

4.5.6 Ordering of memory accesses from GPT walks

I_{CLGHP} Armv8-A [2] includes the following requirement:

> If FEAT_ETS is implemented, and a memory access RW_1 is Ordered-before a second memory access RW_2, then RW_1 is also Ordered-before any translation table walk generated by RW_2 that generates any of the following:
>
> - A Translation fault.
> - An Address size fault.
> - An Access flag fault.

R_{CKTPD} If FEAT_RME is implemented, and a memory access RW_1 is Ordered-before a second memory access RW_2, then RW_1 is also Ordered-before any GPT walk generated by RW_2 that generates any of the following:

- A GPT walk fault.
- A GPT address size fault.

Chapter 5
AArch64 Virtual Memory System Architecture

This section details changes to the AArch64 VMSA.

5.1 Translation regimes

This section introduces:

- Changes to the EL3 translation regime.
- New Realm translation regimes.

I_{FVYPX} For each Security state, configuration of stage 1 and stage 2 translation can produce output addresses only in physical address spaces marked as *YES* in the following table:

	Secure state	Non-secure state	Root state	Realm state
Physical address space				
Secure	Yes	No	Yes	No
Non-secure	Yes	Yes	Yes	Yes
Root	No	No	Yes	No
Realm	No	No	Yes	Yes

5.1.1 Changes to the EL3 translation regime

I_{DFVPL} If translation is enabled, execution at EL3 uses the EL3 stage 1 translation regime.

R_{JFWWV} If translation is disabled, all output addresses from execution at EL3 are Root physical addresses.

R_{CFPDJ} For EL3 stage 1 translation, all levels of lookup are made to the Root physical address space.

R_{XTYPW} For the EL3 stage 1 translation regime, a block or page descriptor fetched from the Root physical address space determines the output physical address space of the translation as described in Table 5.2.

Table 5.2: Output physical address space

TTD.NSE	TTD.NS	Output physical address space
0	0	Secure
0	1	Non-secure
1	0	Root
1	1	Realm

I_{WWBFB} The SCR_EL3.SIF bit has no effect on execution in EL3.

R_{ZWRVD} During execution at EL3, any attempt to execute an instruction fetched from physical memory other than the Root physical address space causes a Permission fault.

See also:

- *5.2 Translation Table descriptor formats*

5.1.2 Realm translation regimes

I_{BXMCL} All translation regimes in Realm state have a mechanism to select if accesses are made to Realm memory or Non-secure memory, at the granularity of the mapping size.

I_{WYXPT} **Realm EL1&0** includes:

Stage 1:

- Two VA ranges.
- Translates VA to Realm IPA.
- Associated with VMID and optionally an ASID.

Stage 2:

- One IPA range.
- Translates Realm IPA to Realm PA or Non-secure PA.
- Associated with a VMID.

Realm EL2&0 includes:

Stage 1:

- Two VA ranges.
- Translates VA to Realm PA or Non-secure PA.
- Optionally associated with an ASID.

Realm EL2 includes:

Stage 1:

- One VA range.
- Translates VA to either Realm PA or Non-secure PA.

I_{MZRPG} Support for execution in Realm state at EL0 in AArch32 is IMPLEMENTATION DEFINED. Use of the Realm translation regimes at EL0 in AArch32 depends on that support for AArch32 at EL0. Support for execution in Realm state at other Exception levels is available in AArch64 only.

See also:

- *3.2 Execution states*

5.1.2.1 Selection of Realm translation regimes

I_{MQGVK} If SCR_EL3.{NSE, NS} selects Secure or Non-secure state, and execution is in EL2 or below, the existing rules from Armv8-A are used to determine the translation regime.

R_{RSTDL} If SCR_EL3.{NSE, NS} selects Realm state and execution is in EL2, and HCR_EL2.E2H == 0, the Realm EL2 stage 1 translation regime is used.

R_{FRCTJ} If SCR_EL3.{NSE, NS} selects Realm state and execution is in EL2, and HCR_EL2.E2H == 1, the Realm EL2&0 stage 1 translation regime is used.

R_{PNNXP} If SCR_EL3.{NSE, NS} selects Realm state and execution is in EL0, and HCR_EL2.{E2H, TGE} == {1, 1}, the Realm EL2&0 stage 1 translation regime is used.

R_{CYRTK} If SCR_EL3.{NSE, NS} selects Realm state and HCR_EL2.TGE == 0, and execution is in EL1 or EL0, the Realm EL1&0 stage 1 and stage 2 translation regime is used.

5.1.2.2 Realm EL1&0 stage 1 translation

R_{TQCYT} Regardless of whether stage 1 translation is enabled or disabled, Realm EL2 is always enabled and output of Realm EL1&0 stage 1 is a Realm IPA for all accesses.

R_{HMWYX} All features supported for the Non-secure EL1&0 stage 1 translation regime are supported for the Realm EL1&0 stage 1 translation regime.

R_{HVZDD} Configuration of SCTLR_EL1, MAIR_EL1, TCR_EL1, TTBR0_EL1, and TTBR1_EL1 has the same effect on Realm stage 1 translation as on Non-secure stage 1 translation.

R_{KHYJK} The Table, Block, and Page descriptors for Realm EL1&0 stage 1 translation have the same format and meaning as for Non-secure stage 1.

5.1.2.3 Realm EL1&0 stage 2 translation

R_{PDRZK} All features supported for the Non-secure EL1&0 stage 2 translations are supported for Realm EL1&0 stage 2 translations.

R_{KCYMF} VTCR_EL2[30:29] are RES0 and there is no equivalent of the NSA, NSW fields for Realm EL1&0 stage 2 translations.

R_{SYSZL} Configuration of HCR_EL2, SCTLR_EL2, VTCR_EL2, and VTTBR_EL2 fields has the same effect on Realm stage 2 translation as on Non-secure stage 2 translation.

R_{PGRQD} All translation table lookups made for Realm EL1&0 stage 2 translation are made to the Realm physical address space.

I_{NVMNC} Realm stage 2 Block and Page descriptors include an NS bit.

R_{LXLSC} If a Block or Page descriptor fetched for Realm EL1&0 stage 2 translation has NS set to 1, the output address is in the Non-secure physical address space. Otherwise, the output address is in the Realm physical address space.

R_{ZFFPX} If Realm EL1&0 stage 2 translation is disabled, accesses to the Realm IPA space are made to the Realm PA space.

R_{QMLYQ} If the stage 2 translation for a Realm stage 1 translation table walk resolves to an address not in the Realm physical address space, it causes a stage 2 Permission fault.

See also:

- *5.2 Translation Table descriptor formats*

5.1.2.4 Realm EL2 stage 1 translation

R_{ZCNMT} Configuration of HCR_EL2, MAIR_EL2, SCTLR_EL2, TCR_EL2, and TTBR0_EL2 has the same effect on Realm EL2 stage 1 translation as on Non-secure EL2 stage 1 translation.

R_{DVGRP} For Realm EL2 stage 1 translation, all levels of lookup are made to the Realm physical address space.

I_{ZLQDZ} Realm EL2 stage 1 Block and Page descriptors include an NS bit.

R_{LYKFZ} If a Block or Page descriptor fetched for Realm EL2 stage 1 translation has NS set to 1, the output address is in the Non-secure physical address space. Otherwise, the output address is in the Realm physical address space.

R_{KVKPM} For execution at Realm EL2, if translation is disabled, all memory accesses are made to the Realm physical address space.

5.1.2.5 Realm EL2&0 stage 1 translation

R_{WGRZN} The determination of output address spaces for Realm EL2&0 stage 1 translation is the same as for Realm EL2 stage 1 translation.

R_{PXWZK} Configuration of HCR_EL2, MAIR_EL2, SCTLR_EL2, TCR_EL2, TTBR0_EL2, and TTBR1_EL2 has the same effect on Realm EL2&0 stage 1 translation as on Non-secure EL2&0 stage 1 translation.

5.1.2.6 Restriction on Realm instruction fetches

R_{PKTDS} If execution is using the Realm EL2 or Realm EL2&0 translation regime, any attempt to execute an instruction fetched from physical memory other than the Realm physical address space causes a stage 1 Permission fault.

R_{HGXXY} If FEAT_PAN3 is implemented, it is IMPLEMENTATION DEFINED whether a stage 1 translation for the Realm EL2&0 translation regime that resolves to a Non-secure address is treated as Unprivileged execute-never for the purpose of PAN.

I_{HFJGN} Permitting an implementation to treat an EL2&0 virtual address that maps to Non-secure physical address as UXN means that the PE does not need to record why the address is not executable when determining whether to permit privileged accesses. This is similar to the interaction between SCR_EL3.SIF and FEAT_PAN3 in Armv8-A [2].

R_{YMCSL} If execution is using the Realm EL1&0 translation regime, any attempt to execute an instruction fetched from physical memory other than the Realm physical address space causes a stage 2 Permission fault.

I_{MQQXW} For the Realm EL1&0 translation regime with stage 2 translation disabled, all output addresses are in the Realm physical address space and therefore Permission faults cannot arise from this mechanism.

5.2 Translation Table descriptor formats

I_{GNQHH} In VMSAv8-64, bit 63 in Table descriptors is RES0 for both stages in Non-secure state, and stage 2 in Secure state.

R_{LZCSQ} For a Table descriptor fetched for stage 1 in the Realm EL2 and Realm EL2&0 translation regimes, bit 63 is RES0 and there is no equivalent of the NSTable field.

R_{TDMTM} For a Table descriptor fetched for stage 1 in the Realm EL1&0 translation regimes, bit 63 is RES0 and there is no equivalent of the NSTable field.

R_{TNYXY} For a Table descriptor fetched for stage 1 in the EL3 translation regime, bit 63 is RES0 and there is no equivalent of the NSTable field.

I_{LPCPS} The removal of NSTable for the EL3 stage 1 translation regime is a change from the behavior of Armv8-A [2].

R_{GYNXY} For a Block or Page descriptor fetched for stage 2 in the Realm Security state, bit 55 is the NS field.

I_{BLLWJ} Bit 55 of stage 2 Block and Page descriptors remains IGNORED for Security states other than Realm Security state.

R_{LWRBF} For a Block or Page descriptor fetched using the EL2 stage 1 or EL2&0 stage 1 translation regimes in the Realm Security state, bit 5 is the NS field.

I_{CZPRF} For a Block or Page descriptor fetched using the EL1&0 stage 1 translation regime in the Realm Security state, bit 5 is RES0.

R_{GVZML} For a Block or Page descriptor fetched using the EL3 stage 1 translation regime, bit 11 is the NSE field.

I_{JRJYP} Bit 11 of stage 1 Block and Page descriptors for translation regimes with two ranges of virtual address space is still the nG bit.

I_{BTPRR} All other bits in the Table, Block, and Page descriptors have the same names and behaviors as described in *About the Virtual Memory System Architecture (VMSA)* in Armv8-A [2].

5.3 TLB maintenance instructions

I_{NZQKX} The rules for TLBI operations are unchanged from Armv8-A [2], apart from being extended to cover the additional Security states as follows:

- TLBI instructions executed in the Realm Security state affect TLB entries inserted for Realm translation regimes.
- TLBI instructions executed at EL3 for a lower Exception level affect TLB entries inserted for translation regimes of the Security state selected by SCR_EL3.{NSE, NS}.

See also:

- 14.3 *TLBI*.

5.4 GPC and hardware management of Access Flag and dirty state

I_{PJWQY} When hardware updates of the Access Flag are enabled, it is permitted to update the Access Flag speculatively. This is not affected by the granule protection check on the output address of the translation.

R_{ZWCSB} For the final enabled stage of translation, when hardware management of dirty state would update a descriptor as part of translating an access, it is permitted to perform the update even if an access to the final output address for that translation experiences a granule protection check fault.

I_{BHPFZ} This is consistent with the requirements for fault reporting priority where stage 1 dirty state can be updated even in the presence of a stage 2 fault on the output address for the stage 1 translation.

See also:

- 3.4.1 *Exceptions from GPC faults*

5.5 Address translation instructions

R_{LMSNQ} Address translation instructions with E0, E1, or E2 are UNDEFINED at EL3 when SCR_EL3.{NSE, NS} == {1, 0}.

I_{MQBPF} SCR_EL3.{NSE, NS} == {1, 0} is Reserved. Therefore, issuing address translation instructions at EL3 for a lower Exception level with SCR_EL3.{NSE, NS} == {1, 0} would be selecting a nonexistent Exception level. This behavior is consistent with how the base architecture handles address translation instructions targeting EL2 when EL2 is not implemented or disabled.

I_{NPRRQ} The following faults are added to the list of faults that can be generated by an address translation instruction:

- GPF.
- GPT address size fault.
- GPT walk fault.
- Synchronous External abort on GPT fetch.

See *Address translation instructions* in Armv8-A [2].

I_{MXTJT} When populating PAR_EL1 with the result of an address translation instruction, granule protection checks are not performed on the final output address of a successful translation. However, granule protection checks are performed on fetches of stage 1 or stage 2 descriptors and these checks could result in a GPC fault.

R_{ZTRDD} In addition to the cases listed in *Address translation instructions* in Armv8-A [2], the following faults as a result of an address translation instruction are reported as an exception:

- GPC faults that would result in a GPC exception.
- GPC faults on fetches of stage 2 descriptors from AT S1E0* and AT S1E1* instructions executed from EL1.
- When HCR_EL2.GPF == 1, GPFs on fetches of stage 1 descriptors from AT S1E0* and AT S1E1* instructions executed from EL1.

Otherwise, faults as a result of an address translation instruction are reported using PAR_EL1.FST.

Chapter 6
Reset

I_{VYYXJ} RME does not modify the behavior of PE registers following a Cold reset or a Warm reset.

Registers that the architecture defines to be UNKNOWN at reset and that may contain Realm sensitive information will be explicitly scrubbed by Root firmware following reset.

I_{CTBBL} RVBAR_EL3 holds the IMPLEMENTATION DEFINED address from which execution starts after reset. At reset, the stage 1 MMU for EL3 is disabled; therefore the address is flat mapped to a PA in the Root physical address space.

R_{XCQVB} A hardware mechanism to reset a PE that is directly exposed to software must only be accessible at EL3.

I_{JYCJY} The Arm architecture guarantees that when EL3 is implemented, any architected control that allows software to reset a PE, for example the Reset Management Register, is only accessible at EL3. This restriction must be applied to all additional IMPLEMENTATION DEFINED reset controls.

Chapter 7
Memory Tagging Extension

I_{BGDTZ} In FEAT_MTE2, instructions that load or store Allocation Tags apply the same address translation and permission checks as a load or store of data to a virtual address.

R_{TQGBX} Accesses to Allocation Tags are subject to a granule protection check on the PA that the Allocation Tags are associated with.

R_{HKVLQ} Fetches of GPT information are Tag Unchecked accesses.

I_{FDMWC} In FEAT_MTE2, it is IMPLEMENTATION DEFINED whether Allocation Tags are permitted to be accessed through regions of the data PA space.

R_{LMRBP} If Allocation Tags are permitted to be accessed through regions of the data PA space, they are only accessible through the Root data PA space.

Chapter 8
Memory partitioning and monitoring

I_ZXBKL Memory Partitioning and Monitoring Extension (MPAM) functionality for a PE that implements FEAT_RME is specified under *Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A* [4].

MPAM defines independent PARTID spaces for Non-secure and Secure states distinguished by the MPAM_NS attribute. MPAM for RME replaces MPAM_NS with a 2-bit MPAM_SP (*MPAM Space*) attribute that allows Arm CCA systems to implement four independent PARTID spaces, one for each Security state. It also defines how multiple Security states might share a single PARTID space.

I_PWTXR A PE that implements RME is capable of generating a 2-bit MPAM_SP encoded as:

- `0b00`, Secure.
- `0b01`, Non-secure.
- `0b10`, Root.
- `0b11`, Realm.

R_PNBCB GPT accesses as a result of a data access or translation table walk use PARTID_D and PMG_D for the current Exception level and Security state.

R_SZYNF GPT accesses as a result of an instruction fetch use PARTID_I and PMG_I for the current Exception level and Security state.

Chapter 9
RAS

This section covers requirements relating to the use of the Arm RAS architecture [5].

A key requirement on RAS support for Arm CCA is to maintain the security isolation boundaries, of confidentiality and integrity, provided by RME. A challenge with this requirement is that there is strong market desire for RAS to be triaged in the Non-secure state, in hypervisor or kernel code. This implies PEs running in Non-secure state having direct access to sanitized Error Record registers.

While the rules written in this chapter are in terms of the Arm RAS architecture, the same concerns on the ability to observe or modify confidential information, or the ability to control reporting of RAS events, apply to proprietary RAS solutions.

9.1 Confidential information in RAS Error Records

R_{QGXBC} For the purposes of this section, *confidential information* is defined as information that is not accessible to the current Security state under normal operation. This information comprises:

- Values of memory locations to which accesses are prohibited by the programming of the Granule Protection Table.
- Values of general-purpose registers, SIMD, SVE, or System registers with context associated with execution from a different Security state.

Memory content in the Root physical address space, or execution context from the Root Security state is considered Root confidential information.

Memory content in the Secure physical address space, or execution context from the Secure Security state is considered Secure confidential information.

Memory content in the Realm physical address space, or execution context from the Realm Security state is considered Realm confidential information.

I_{DWNBJ} Confidential information, depending on which Security state a PE is executing in, follows the rules relating to Security state and physical address space described in 4.1 *Physical address spaces*. That is:

- When executing in Root Security state, there is no confidential information.
- When executing in Secure state, Root and Realm information is confidential.
- When executing in Realm state, Root and Secure information is confidential.
- When executing in Non-secure state, Root, Secure, and Realm information is confidential.

The following are not considered to be confidential information:

- The address at which a given error is detected and that is returned in ERR<n>ADDR. There are exceptions in the case of the error injection, see 9.4 *RAS Error injection*.
- Information that identifies a *field replaceable unit* (FRU) and that is returned in ERR<n>MISC registers.
- Information used to ascertain the severity of an error, such as:
  - Error record status information in ERR<n>STATUS.
  - Error counters.
- Information used to ascertain the properties of an error node, identification, and affinity.

I_{QVYQR} RAS error record accesses from the Non-secure Security state do not expose Secure, Realm, or Root confidential information.

I_{VRSJM} RAS error record accesses from the Secure Security state do not expose Realm or Root confidential information.

I_{HJZRH} RAS error record accesses from the Realm Security state do not expose Secure or Root confidential information.

I_{DWGGW} A number of implementation options are possible:

- RAS error record registers contain no confidential information, for example by only providing address, FRU information and severity information.
- The data provided by RAS error record, that operates on data belonging to more than one physical address space or Security state, depends on the Security state of a Requester making access, such that any confidential information is removed from the record.
- RAS error record registers that might contain confidential information are only accessible in the Root Security state. This means:
  - Memory mapped RAS error record registers are only accessible in the Root physical address space.
  - For RAS error record register exposed through System registers, Root firmware can use SCR_ELR.TERR to restrict access to only the Root Security state.

Arm strongly recommends against making error records only accessible in the Root Security state.

9.2 RAS Error detection and correction

I_{KPWLS} A less secure state must not be able to disable error correction and detection of accesses made from a more secure state.

R_{SXKNQ} The following applies to RAS error detection and correction capabilities:

- RAS error detection and correction on resources that might be accessed in the Root Security state, can not be disabled using RAS controls accessible in the Realm, Secure, or Non-secure physical address space, or by PEs in the Realm, Secure, or Non-secure Security state.
- RAS error detection and correction on resources that might be accessed in the Secure Security state, can not be disabled using RAS controls accessible in the Realm, or Non-secure physical address space, or by PEs in the Realm, or Non-secure Security state.
- RAS error detection and correction on resources that might be accessed in the Realm Security state, can not be disabled using RAS controls accessible in the Secure, or Non-secure physical address space, or by PEs in the Secure, or Non-secure Security state.

9.3 RAS Error signaling

The following rules describe constraints on how errors must be signaled to PEs in an RME enabled system using the Arm RAS architecture.

R_{VKZNJ} Error signaling and recording controls related to RAS error records that might contain confidential information are only accessible in the Root Security state.

I_{KNTKK} The Armv8-A RAS specification [5] requires that an error exception must be generated for all detected errors that are signaled to and consumed by a PE as an External abort in response to an architectural read. For PEs implementing FEAT_RME, GPT fetches are considered architectural reads.

I_{YDRLJ} As described in 4.5.2 *GPC faults*, RAS errors detected on GPT fetches to check accesses cause synchronous External abort at Level x faults. As described in 3.4.2 *Granule protection check exceptions*, these faults are reported to the PE using GPC exceptions, which are synchronous. This is different from translation table fetches, where it is IMPLEMENTATION DEFINED whether they are taken synchronously or asynchronously.

See also:

- 3.4.2 *Granule protection check exceptions*
- 4.5.2 *GPC faults*
- 15.1.5 *ESR_ELx*

9.4 RAS Error injection

The rules in this section apply to the *Common Fault Injection Model Extension*, an optional part of RAS System Architecture v1.1 [5], or equivalent IMPLEMENTATION DEFINED fault injection controls.

R_{XGXNW} It must not be possible to determine which address a PE is accessing, through the address reported in ERR<n>ADDR as a result of an injected fault, when the following are true:

- The ERR<n>ADDR register is accessible to PEs not in the Root Security state.
- The controls for fault injection are available to PEs not in the Root Security state.

I_{FRCGL} An implementation can approach this in number of different ways, for example:

- Not update the ERR<n>ADDR as a result of the injection.
- Not provide a valid address.

I_{CMMRJ} Exposing the address would allow a less secure agent to determine code execution of the PE when it executes in a more secure state.

R_{NYGDN} A fault injection model which results in an error being signaled when a PE accesses a specific physical address, must not be implemented when the following are true:

- The address can be set by a PE not in the Root Security state.
- The controls for the fault injection are available to PEs not in the Root Security state.

I_{HGSSL} The Common Fault Injection Model Extension of the RAS System Architecture v1.1 [5] is compliant with this rule, as it does not support the signaling of errors when an agent accesses a specific address.

I_{RJKDQ} Data is not corrupted by the Common Fault Injection Model Extension of the RAS System Architecture v1.1 [5].

R_{ZBSKS} IMPLEMENTATION DEFINED error injection mechanisms do not corrupt data stored at memory locations where errors are injected.

Chapter 10
AArch64 Self-hosted Debug and Trace

This section details changes to the AArch64 self-hosted debug and self-hosted trace support.

10.1 Self-hosted debug

I_{MLZKF} RME makes no changes to the existing routing and trap controls for self-hosted debug. Existing EL1 and EL2 controls are described in terms of the *current Security state*, and naturally extend to the states that are added by RME.

10.1.1 Execution conditions for watchpoints and breakpoints

I_{MBPRS} Each watchpoint or breakpoint can be programmed so that it only generates exceptions for certain execution conditions. For example, a watchpoint might be programmed to generate Watchpoint exceptions only when the PE is executing at EL2 in Non-secure state. RME adds an additional field, SSCE, to DBGWCR<n>_EL1 and DBGBCR<n>_EL1. Together with the existing SSC, HMC, and PxC fields, these control when the watchpoint or breakpoint can trigger.

R_{PNDXR} The SSCE, SSC, HMC, and PAC fields in DBGWCR<n>_EL1 define the execution conditions when a watchpoint triggers. Similarly, the SSCE, SSC, HMC, and PMC fields in DBGBCR<n>_EL1 define the execution conditions when a breakpoint triggers.

The permitted combinations are shown here:

HMC	SSCE	SSC	PxC	Security state	EL3	EL2	EL1	EL0
0	0	00	01	S, NS, RL	-	-	Y	-
0	0	00	10	S, NS, RL	-	-	-	Y
0	0	00	11	S, NS, RL	-	-	Y	Y
0	0	01	01	NS	-	-	Y	-
0	0	01	10	NS	-	-	-	Y
0	0	01	11	NS	-	-	Y	Y
0	0	10	01	S	-	-	Y	-
0	0	10	10	S	-	-	-	Y
0	0	10	11	S	-	-	Y	Y
0	0	11	00	S	-	Y	-	-
0	0	11	01	S	-	Y	Y	-
0	0	11	11	S	-	Y	Y	Y
1	0	00	01	S, NS, RL, RT	Y	Y	Y	-
1	0	00	11	S, NS, RL, RT	Y	Y	Y	Y
1	0	01	00	NS	-	Y	-	-
1	0	01	01	NS	-	Y	Y	-
1	0	01	11	NS	-	Y	Y	Y
1	0	10	00	RT	Y	-	-	-
1	0	10	01	S, RT	Y	Y	Y	-
1	0	10	11	S, RT	Y	Y	Y	Y
1	0	11	00	S, NS, RL	-	Y	-	-

HMC	SSCE	SSC	PxC	Security state	EL3	EL2	EL1	EL0
1	0	11	01	S, NS, RL	-	Y	Y	-
1	0	11	11	S, NS, RL	-	Y	Y	Y
0	1	01	01	RL	-	-	Y	-
0	1	01	10	RL	-	-	-	Y
0	1	01	11	RL	-	-	Y	Y
1	1	01	00	RL	-	Y	-	-
1	1	01	01	RL	-	Y	Y	-
1	1	01	11	RL	-	Y	Y	Y

Where:

- NS is Non-secure state.
- S is Secure state.
- RL is Realm state.
- RT is Root state.
- PxC is PMC or PAC as appropriate.

All combinations of HMC, SSCE, SSC, and PxC that this table does not show are reserved.

I_{LVPDJ} For a PE that does not implement FEAT_RME, SSCE is RES0.

10.2 Self-hosted trace

I_{RRZMV} A PE that implements FEAT_RME can optionally implement the Trace Architecture[1]. If the Trace Architecture is implemented, the following extensions are also implemented:

- FEAT_TRF.
- FEAT_TRBE.

I_{YWRJK} RME makes no changes to the existing EL1 and EL2 trap controls for self-hosted trace. Existing EL1 and EL2 trap controls are described in terms of the *current Security state*, and naturally extend to the states that are added by RME.

I_{MRZJT} The MDCR_EL3.NSTB field is extended to cover the Realm Security state.

10.2.1 Register controls to enable self-hosted trace

R_{NMCQK} If FEAT_TRF is implemented, self-hosted trace is enabled if one of the following is true:

- EDSCR.TFO == 0.
- EDSCR.TFO == 1, EL3 is not implemented, the PE executes in Secure state and `ExternalSecureNoninvasiveDebugEnabled()` == FALSE.
- EDSCR.TFO == 1, EL3 is implemented, MDCR_EL3.STE == 1 and `ExternalSecureNoninvasiveDebugEnabled()` == FALSE.
- EDSCR.TFO == 1, FEAT_RME is implemented, MDCR_EL3.RLTE == 1 and `ExternalRealmNoninvasiveDebugEnabled()` == FALSE.

10.2.2 Prohibited regions in trace

R_{ZPJRJ} If `SelfHostedTraceEnabled()` == FALSE, tracing is prohibited in Root state when `ExternalRootNoninvasiveDebugEnabled()` == FALSE.

R_{RJCNQ} If `SelfHostedTraceEnabled()` == FALSE, tracing is prohibited in Realm state when `ExternalRealmNoninvasiveDebugEnabled()` == FALSE.

10.2.3 Trace buffer management

I_{XFDFB} Writes to the trace buffer are subject to granule protection checks and might trigger GPC faults. These are reported as Trace buffer management events, in the same way that VMSA faults are reported.

R_{SBYYT} A write to the trace buffer that triggers a GPF is reported as a trace buffer management event with the following syndrome:

Access that triggers GPF	TRBSR_EL1.EC
Stage 1 walk or table update	`0b100100`
Stage 2 walk or table update	`0b100101`
Write to Trace Buffer	`0b100100`

R_{RKQTS} A write to the trace buffer that triggers any of the following is reported as a trace buffer management event with TRBSR_EL1.EC == `0b01_1110`:

- GPT address size fault.

- GPT walk fault.
- Synchronous External abort on GPT fetch.

Chapter 11
External debug and trace

This section covers changes to external debug features.

11.1 Architecture extensions

I_{RHHXW} A PE that implements FEAT_RME also implements the following architecture extensions:

- FEAT_DoPD.
- FEAT_Debugv8p4.

I_{PWBMD} A PE that implements FEAT_RME can optionally support the PC Sample-based Profiling Extension. If the PC Sample-based Profiling Extension is implemented, the following extensions are supported:

- FEAT_PCSRv8.
- FEAT_PCSRv8p2.

I_{MMTVW} A PE that implements FEAT_RME can optionally support the Trace Architecture[1]. If the Trace Architecture is implemented, the following extension is supported:

- FEAT_ETEv1p2.

11.2 Required debug authentication

I_LGVDQ RME provides additional external debug authentication for Realm and Root states.

R_VSRBC For a PE that implements FEAT_RME, the following additional debug authentication pseudocode functions are defined:

Pseudocode function	Description
`ExternalRootInvasiveDebugEnabled()`	Returns TRUE if Root invasive debug is enabled
`ExternalRootNoninvasiveDebugEnabled()`	Returns TRUE if Root non-invasive debug is enabled
`ExternalRealmInvasiveDebugEnabled()`	Returns TRUE if Realm invasive debug is enabled
`ExternalRealmNoninvasiveDebugEnabled()`	Returns TRUE if Realm non-invasive debug is enabled

I_KMJWJ These are equivalent to the existing functions for Secure and Non-secure state.

R_KNNKM The following conditions always apply:

- If `ExternalInvasiveDebugEnabled()` == FALSE then `ExternalRealmInvasiveDebugEnabled()` == FALSE.
- If (`ExternalInvasiveDebugEnabled()` && `ExternalSecureInvasiveDebugEnabled()` && `ExternalRealmInvasiveDebugEnabled()`) == FALSE then `ExternalRootInvasiveDebugEnabled()` == FALSE.
- `ExternalRealmNoninvasiveDebugEnabled()` returns the same as `ExternalRealmInvasiveDebugEnabled()`.
- `ExternalRootNoninvasiveDebugEnabled()` returns the same as `ExternalRootInvasiveDebugEnabled()`.

I_CPKJM This is in addition to the conditions defined in section *Required debug authentication* of Armv8-A [2].

11.2.1 Recommended authentication signals

I_PSNRD The details of the debug authentication interface are IMPLEMENTATION DEFINED, but Arm recommends the following additional signals for external debug authentication:

- **RLPIDEN**.
- **RTPIDEN**.

I_KBYRS The recommended mapping between the authentication pseudocode functions and the signals is:

Pseudocode function	Implementation
`ExternalRootInvasiveDebugEnabled()`	**DBGEN** AND **RLPIDEN** AND **SPIDEN** AND **RTPIDEN**
`ExternalRealmInvasiveDebugEnabled()`	**DBGEN** AND **RLPIDEN**
`ExternalSecureInvasiveDebugEnabled()`	**DBGEN** AND **SPIDEN**

I_BGYHS In order to include External debug state in Realm attestation, the authentication signals that control External debug in the Root and Realm Security states must be sampled before execution starts in the corresponding Security state and not change value until a system reset event.

For the RTPIDEN authentication signal, Arm expects that this behavior will be guaranteed by system construction.

For the RLPIDEN authentication signal, this behavior can be guaranteed by system construction or by Root firmware.

I$_{GTFYJ}$ This table shows the Security states externally debuggable with different values for the debug authentication signals, based on the recommended mapping:

DBGEN	**RLPIDEN**	**SPIDEN**	**RTPIDEN**	Non-secure	Realm	Secure	Root
0	x	x	x	No	No	No	No
1	0	0	x	Yes	No	No	No
1	1	0	x	Yes	Yes	No	No
1	0	1	x	Yes	No	Yes	No
1	1	1	0	Yes	Yes	Yes	No
1	1	1	1	Yes	Yes	Yes	Yes

11.3 Halting allowed and halting prohibited

R_{KWYXV} In addition to the rules in section *Halting allowed and halting prohibited* of Armv8-A [2], halting is prohibited when:

- The PE is in Realm state and `ExternalRealmInvasiveDebugEnabled()` == FALSE.
- The PE is in Root state and `ExternalRootInvasiveDebugEnabled()` == FALSE.

11.4 Imprecise entry to Debug state

I_{HKLSW} Entry to Debug state is normally precise, meaning that the PE can not enter Debug state if it can neither complete nor abandon all currently executing instructions and leave the PE in a precise state. The architecture has OPTIONAL support for imprecise entry to Debug state through a debugger writing to EDRCR.CBRRQ.

R_{RVNFD} If imprecise entry to Debug state is supported, writes to EDRCR.CBRRQ are ignored when:

- `ExternalInvasiveDebugEnabled()` == FALSE.
- FEAT_RME is implemented and `ExternalRootInvasiveDebugEnabled()` == FALSE.
- FEAT_RME is not implemented, `ExternalSecureInvasiveDebugEnabled()` == FALSE, and either:
  - EL3 is not implemented and the implemented Security state is Secure state.
  - EL3 is implemented.

11.5 Summary of actions from debug events

R_{SCDBZ} In Armv8-A [2], the meaning of Authentication in Table H2-1 (*Debug authentication for external debug*) is extended to cover:

- `ExternalnvasiveDebugEnabled()`.
- `ExternalRealmInvasiveDebugEnabled()`.
- `ExternalSecureInvasiveDebugEnabled()`.
- `ExternalRootInvasiveDebugEnabled()`.

11.6 Controlling the PC Sample-based Profiling Extension

R_{MHHZD} PC sample-based Profiling is prohibited unless both:

- `ExternalNoninvasiveDebugEnabled()` == TRUE.
- At least one of the following applies:
  - The PE is executing in Non-secure state.
  - EL3 is not implemented.
  - EL3 is implemented, the PE is executing in Secure state, and `ExternalSecureNoninvasiveDebugEnabled()` == TRUE.
  - The PE is executing in Realm state and `ExternalRealmNoninvasiveDebugEnabled()` == TRUE.
  - The PE is executing in Root state and `ExternalRootNoninvasiveDebugEnabled()` == TRUE.

11.7 ETE

I_{SDZHN} This section describes changes to ETE. The base ETE specification is described in Armv9-A [1].

R_{CQNGM} For the following ETE packets:

- Exception 32-bit address IS0 with Context packet.
- Exception 32-bit address IS1 with Context packet.
- Exception 64-bit address IS0 with Context packet.
- Exception 64-bit address IS1 with Context packet.
- Context packet.
- Target address with Context 32-bit IS0 packet.
- Target address with Context 32-bit IS1 packet.
- Target address with Context 64-bit IS0 packet.
- Target address with Context 64-bit IS1 packet.

In the byte that contains the NS field, a new field is allocated:

Bit [3] NSE

Together with the NS field, this field reports the Security state.

R_{GXRQT} For ETE packets with NS and NSE fields, the NS and NSE fields report the Security state:

NSE	NS	Meaning
0	0	Secure
0	1	Non-secure
1	0	Root
1	1	Realm

I_{ZQNLR} TRCVICTLR and TRCACATR<n> are extended with fields for Realm Exception levels.

R_{GXWWX} The ETE exception type used for indicating a GPC exception is either an Inst Fault or a Data Fault, based on the type of the access that triggered the GPC exception.

Chapter 12
Performance Profiling

This section covers changes to Performance Monitors Unit (PMU) and Statistical Profiling Extension (SPE).

12.1 Performance Monitors

I_{BQSPH} PME is an OPTIONAL feature of an implementation, but Arm strongly recommends that implementations include version 3 of the Performance Monitors Extension (PMUv3), or later.

If the Performance Monitors Extension is implemented, the following extension is implemented:

- FEAT_PMUv3p7.

I_{MVCKW} RME makes no changes to the existing routing and trap controls for the Performance Monitors. Existing EL1 and EL2 controls are described in terms of the *current Security state*, therefore naturally extend to the states added by RME.

R_{MNKRS} All existing architectural TLB-related PMU events are permitted but not required to count GPT accesses, GPT-related TLB hits, or GPT-related TLB misses, for the current Security state, as appropriate to the nature of the PMU event.

R_{LMLSR} IMPLEMENTATION DEFINED events are permitted to count GPT accesses, GPT-related TLB hits, or GPT-related TLB misses for the current Security state.

I_{YMTVH} FEAT_MTPMU [2] is an OPTIONAL feature.

I_{MGZPQ} If FEAT_MTPMU is implemented, in production environments, EL3 software is expected to set MDCR_EL3.MTPME to 0, meaning that PMEVTYPERn_EL0.MT is RES0. The status of multi-threaded PMU support is expected to be indicated by Realm attestation.

12.2 Statistical profiling

I_{GMPZP} A PE that implements FEAT_RME can optionally implement the Statistical Profiling Extension. If the Statistical Profiling Extension is implemented, the following extension is also implemented:

- FEAT_SPEv1p2.

I_{NQJWW} RME makes no changes to the existing EL1 and EL2 trap controls for statistical profiling.

I_{RGKRK} RME extends the MDCR_EL3.NSPB trap to cover the Realm Security state.

12.2.1 Profiling Buffer management

I_{SPZXX} Writes to the Profiling Buffer are subject to granule protection checks and might trigger GPC faults. These are reported as Profiling Buffer management events, in the same way that VMSA faults are reported.

R_{RMCBP} A write to the Profiling Buffer that triggers a GPF is reported as a Profiling Buffer management event with the following syndrome:

Access that triggered GPF	PMBSR_EL1.EC
Stage 1 walk or table update	0b100100
Stage 2 walk or table update	0b100101
Write to Profiling Buffer	0b100100

R_{JXVJD} A write to the Profiling Buffer that triggers any of:

- GPT address size fault.
- GPT walk fault.
- Synchronous External abort on GPT fetch.

Is reported as a Profiling Buffer management event with PMBSR_EL1.EC == 0b01_1110.

See also:

- 15.1.18 *PMBSR_EL1*

12.2.2 Statistical profiling extension sample records

R_{BQBMQ} For *Address packet payload* bit assignments, when the format is *Data access physical address*, a new field is assigned to byte 7:

Bit [4] NSE

Together with the NS field, this field reports the physical address space.

R_{YQCDG} For *Address packet payload* bit assignments, when the format is *Data access physical address*, collectively the NS and NSE fields report the physical address space:

NSE	NS	Meaning
0	0	Secure
0	1	Non-secure

NSE	NS	Meaning
1	1	Realm

All other encodings are reserved.

I_{WPMNH} There is no encoding for Root as SPE sample records are not generated by EL3.

R_{YFPDW} For *Address packet payload* bit assignments, when the format is *Instruction virtual address*, a new field is assigned to byte 7:

Bit [4] NSE

Together with the NS field, this field reports the Security state associated with the address.

R_{LPRNF} For *Address packet payload* bit assignments, when the format is *Instruction virtual address*, collectively the NS and NSE fields report the Security state associated with the address:

NSE	NS	Meaning
0	0	Secure
0	1	Non-secure
1	1	Realm

I_{YBDVB} There is no encoding for Root as SPE sample records are not generated by EL3.

12.3 Branch Record Buffer Extension

I_{CYGYG} The Branch Record Buffer Extension (BRBE) is described in Armv9-A [1].

I_{DJSGX} RME makes no changes to BRBE.

Chapter 13
Performance Management

This section covers changes to Activity Monitors Unit.

I_{ZGDTB} The Activity Monitors Extension is an OPTIONAL extension.

I_{MZPLX} RME makes no changes to the existing trap controls for the Activity Monitors.

I_{RZZJN} Arm strongly recommends that, where implemented, the Activity Monitors Extension includes FEAT_AMUv1p1.

I_{NJBQH} The Activity Monitors Extension implements two counter groups:

- A counter group of four architected 64-bit event counters. The events counted by the counter group are fixed and architecturally defined. These events are:
  - CPU_CYCLES, Processor frequency cycles.
  - CNT_CYCLES, Constant frequency cycles.
  - INST_RETIRED, Instructions retired.
  - STALL_BACKEND_MEM, Memory stall cycles.
- Auxiliary event counters count events defined by the Performance Monitors architecture and IMPLEMENTATION DEFINED events designed specifically for activity monitoring.

Arm recommends that CNT_CYCLES and CPU_CYCLES are not context switched or disabled when Realms are scheduled. These counters are commonly used for performance management in commonly used operating systems. CNT_CYCLES count can be ascertained by Non-secure supervisory software by observing passage of time of the generic counter. CPU_CYCLES can be similarly derived if the software has access to the current frequency of the PE, which is generally available to operating system kernels.

Realm attestation is expected to report potential usage of Activity Monitors Extension by Non-secure Exception levels, to measure PE activity while executing in Realm Security state.

The architecture permits a number of programming models in the presence of Realms:

Disablement

If FEAT_AMUv1p1 is implemented, EL3 can disable visibility of all auxiliary counters to lower Exception levels by using AMCR_EL0.CG1RZ. Software running at EL3 and entities using the memory mapped view, such a trusted power controller, can still observe counting of auxiliary counters.

EL3 can enable or disable any individual counter using AMCNTENCLR0_EL0.

Content switching

If FEAT_AMUv1p1 is implemented, EL2 virtual offsets can be context switched. The offsets, AMEVCNTVOFF0<n>_EL2, are available for all counters except for CNT_CYCLES.

EL3 can also stop the counters, write a new value, and resume.

Chapter 14
AArch64 instructions

This section covers changes to A64 instructions. For full information on System instructions, see Chapter A3 *List of instructions*.

14.1 SVE and SVE2

R_{HJTVJ} For data accesses as a result of an SVE Non-fault load, GPC faults are treated consistently with stage 1 and 2 MMU faults.

R_{SNZTS} For data accesses as a result of an SVE First-fault load where the First active element does not cause a fault, GPC faults for the other elements are treated consistently with stage 1 and 2 MMU faults.

I_{QCSRH} SVE Non-fault loads suppress exceptions due to faults from Active elements, instead setting predicate bits in the FFR to indicate how much data was successfully loaded. Similarly, SVE First-fault loads suppress exceptions due to faults on all but the first Active element.

I_{VKDKF} First-fault and Non-fault loads are defined in the SVE specification [6].

14.2 CFP RCTX, CPP RCTX, and DVP RCTX

R_{QYYHJ} In the register argument to the CFP RCTX, CPP RCTX and DVP RCTX instructions, a new field is added:

Bit [27] NSE

Together with the NS field, selects the Security state.

This field is RES0 in PEs that do not implement FEAT_RME.

R_{HBJQH} In the register argument to the CFP RCTX, CPP RCTX and DVP RCTX instructions, the description of the NS field is amended to:

Bit [26] NS

Together with the NSE field, selects the Security state.

R_{LZTWP} For CFP RCTX, CPP RCTX, and DVP RCTX instructions, collectively the NS and NSE fields in the register argument select the Security state:

NSE	NS	Meaning
0	0	Secure
0	1	Non-secure
1	0	Root
1	1	Realm

When executed in Secure state, the Effective value of NSE is 0.

When executed in Non-secure state, the Effective value of {NSE, NS} is {0, 1}.

When executed in Realm state, the Effective value of {NSE, NS} is {1, 1}.

R_{PMCNZ} If a CFP RCTX, CPP RCTX, or DVP RCTX System instruction is executed at EL3, with {NSE, NS} == {1, 0} and with the EL field value being other than `0b11` (EL3), then the instruction is treated as a NOP.

See also:

- *3.3 Security states*

14.3 TLBI

I_{RKPFZ} In the Armv8-A architecture, the arguments to TLBI operations are for the input to the translation stage they target.

R_{HWLWM} In TLBI operation definitions that include the phrase:

When EL2 is implemented and enabled in the current Security state

The phrase is replaced with:

When EL2 is implemented and enabled in the Security state selected by SCR_EL3.{NSE, NS}

R_{MZSMG} For TLBI operations with E1 or E2, the determination of which entry or entries are required to be invalidated is extended to cover Realm state. The subbullet points relating to SCR_EL3.NS are removed, and replaced with:

- SCR_EL3.{NSE, NS} == {0, 0} and the entry would be required to translate the appropriate Secure translation regime.
- SCR_EL3.{NSE, NS} == {0, 1} and the entry would be required to translate the appropriate Non-secure translation regime.
- SCR_EL3.{NSE, NS} == {1, 1} and the entry would be required to translate the appropriate Realm translation regime.

R_{CQCBH} Executing a TLBI instruction with E1 or E2, while SCR_EL3.{NSE, NS} == {1, 0}, is not required to invalidate any TLB entries.

I_{RMHFH} The Root Security state has no EL2 or stage 2 translation. SCR_EL3.{NSE, NS} == {1, 0} is reserved, and does not select Root state for TLBIs.

I_{YZGXZ} TLBI operations with E3 always operate on the EL3 translation regime.

R_{GRMNJ} In the register argument to:

- TLBI IPAS2E1.
- TLBI IPAS2E1IS.
- TLBI IPAS2E1OS.
- TLBI IPAS2LE1.
- TLBI IPAS2LE1IS.
- TLBI IPAS2LE1OS.
- TLBI RIPAS2E1.
- TLBI RIPAS2E1IS.
- TLBI RIPAS2E1OS.
- TLBI RIPAS2LE1.
- TLBI RIPAS2LE1IS.
- TLBI RIPAS2LE1OS.

The description of the NS is extended to cover Realm state:

Bit [63] NS

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0	IPA is in the Secure IPA space.
1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

I_{TCCYV} The NS field is not needed for TLBI IPA operations targeting Realm state, as Realm state has only a single IPA space.

I_{CXSWQ} If FEAT_RME is not implemented, then when SCR_EL3.NS == 0 and SCR_EL3.EEL2 == 0, the following TLBI operations have no effect when executed at EL3:

- TLBI IPAS2E1.
- TLBI IPAS2E1IS.
- TLBI IPAS2E1OS.
- TLBI IPAS2LE1.
- TLBI IPAS2LE1IS.
- TLBI IPAS2LE1OS.
- TLBI RIPAS2E1.
- TLBI RIPAS2E1IS.
- TLBI RIPAS2E1OS.
- TLBI RIPAS2LE1.
- TLBI RIPAS2LE1IS.
- TLBI RIPAS2LE1OS.

If FEAT_RME is implemented, then the equivalent condition for these operations having no effect when executed at EL3 is SCR_EL3.{NSE, NS} == {0, 0} and SCR_EL3.EEL2 == 0.

Chapter 15
AArch64 PE architectural state

This section covers changes to PE state. For full information of registers, see Chapter A2 *List of registers*.

15.1 System registers

This section covers changes to Special and System registers.

15.1.1 CNTHCTL_EL2

R_{HPNXF} CNTHCTL_EL2[19] is allocated as CNTPMASK:

Value	Meaning
0b0	This control has no affect on CNTP_CTL_EL0.IMASK.
0b1	CNTP_CTL_EL0.IMASK behaves as if set to 1 for all purposes other than a direct read of the field.

This bit is RES0 in Non-secure and Secure state.

This field resets to an architecturally UNKNOWN value.

This field is allocated when HCR_EL2.E2H == 1 and when HCR_EL2.E2H == 0.

This field is RES0 in PEs that do not implement FEAT_RME.

R_{MXJRX} CNTHCTL_EL2[18] is allocated as CNTVMASK:

Value	Meaning
0b0	This control has no effect on CNTV_CTL_EL0.IMASK.
0b1	CNTV_CTL_EL0.IMASK behaves as if set to 1 for all purposes other than a direct read of the field.

This bit is RES0 in Non-secure and Secure state.

This field resets to an architecturally UNKNOWN value.

This field is allocated when HCR_EL2.E2H == 1 and when HCR_EL2.E2H == 0.

This field is RES0 in PEs that do not implement FEAT_RME.

15.1.2 DBGAUTHSTATUS_EL1

See 15.3.4 *DBGAUTHSTATUS_EL1*.

15.1.3 DBGBCR<n>_EL1

R_{LPVHH} DBGBCR<n>_EL1[29] is allocated as the Security State Control Extended (SSCE) field:

Together with the SSC, HMC and PAC fields, this field determines the Security states under which a Breakpoint debug event for breakpoint n is generated.

15.1.4 DBGWCR<n>_EL1

R_{VYQCR} DBGWCR<n>_EL1[29] is allocated as the Security State Control Extended (SSCE) field:

Together with the SSC, HMC and PAC fields, this field determines the Security states under which a Watchpoint debug event for watchpoint n is generated.

15.1.5 ESR_ELx

15.1.5.1 ISS encoding for an exception from a Data or Instruction Abort

R_{MDGDM} In the ISS encodings *ISS encoding for an exception from a Data Abort*, and *ISS encoding for an exception from an Instruction Abort*, the following additional *xFSC* encodings are defined:

xFSC	Meaning
`0b100011`	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.
`0b1001xx`	Granule Protection Fault on translation table walk or hardware update of translation table, level *xx*.
`0b101000`	Granule Protection Fault not on translation table walk or hardware update of translation table.

The encoding `0b100011` is only present if FEAT_LPA2 is implemented.

I_{VVFBP} The other fields in *ISS encoding for an exception from a Data Abort* and *ISS encoding for an exception from an Instruction Abort* are unchanged.

15.1.5.2 EC and ISS encoding for an exception from a Granule Protection Check

R_{KJGHJ} ESR_ELx.EC == `0b01_1110` is allocated as *Granule Protection Check exception* and uses the ISS encoding *ISS encoding for an exception from a Granule Protection Check.*

R_{WPNQF} *ISS encoding for an exception from a Granule Protection Check* is defined as:

Bits [24:22] RES0

Bits [21] S2PTW

Indicates whether the Granule Protection Check exception was on an access made for a stage 2 translation table walk:

S2PTW	Meaning
`0b0`	Fault not on a stage 2 translation table walk.
`0b1`	Fault on stage 2 translation table walk.

This field resets to an architecturally UNKNOWN value.

Bits [20] InD

Indicates whether the Granule Protection Check exception was on an instruction or data access.

InD	Meaning
`0b0`	Data access.
`0b1`	Instruction access.

This field resets to an architecturally UNKNOWN value.

Bits [19:14] GPCSC

Granule Protection Check Status Code

GPCSC	Meaning
`0b000000`	GPT address size fault, at GPT Level 0.
`0b00010x`	GPT walk fault, at GPT Level *x*.
`0b00110x`	Granule Protection Fault, at GPT Level *x*.
`0b01010x`	Synchronous External abort on GPT fetch, at GPT Level *x*.

All other values are reserved.

This field resets to an architecturally UNKNOWN value.

Bits [13] VNCR

Same definition as in *ISS encoding for an exception from a Data Abort*.

When InD==1, this field is RES0.

This field resets to an architecturally UNKNOWN value.

Bits [12:11] RES0

Bits [10:9] RES0

Bits [8] CM

Same definition as in *ISS encoding for an exception from a Data Abort*.

This field resets to an architecturally UNKNOWN value.

Bits [7] S1PTW

Indicates whether the Granule Protection Check exception was on an access for stage 2 translation for a stage 1 translation table walk:

Same encoding as in *ISS encoding for an exception from a Data Abort*.

This field resets to an architecturally UNKNOWN value.

Bits [6] WnR

Same definition as in *ISS encoding for an exception from a Data Abort*.

When InD==1, this field is RES0.

This field resets to an architecturally UNKNOWN value.

Bits [5:0] xFSC

Instruction or Data Fault Status Code

Same definition as in *ISS encoding for an exception from a Data Abort*.

This field resets to an architecturally UNKNOWN value.

I_{BXZXQ} Collectively the xFSC, S1PTW and S2PTW fields in the GPC exception syndrome report whether the GPC fault was on a translation table walk.

xFSC	S1PTW	S2PTW	Meaning
101000	0	0	GPC not on translation table walk.
1001xx	0	0	GPC on stage 1 translation table walk at level *xx*.
1001xx	0	1	GPC on stage 2 translation table walk at level *xx*, not as part of a stage 1 translation table walk.
1001xx	1	1	GPC on stage 2 translation table walk at level *xx*, as part of a stage 1 translation table walk.
100011	0	0	GPC on stage 1 translation table walk at level -1.
100011	0	1	GPC on stage 2 translation table walk at level -1, not as part of a stage 1 translation table walk.
100011	1	1	GPC on stage 2 translation table walk at level -1, as part of a stage 1 translation table walk.

15.1.6 ID_AA64ISAR0_EL1

I_{PGWMW} The value returned by a direct read of ID_AA64ISAR0_EL1.RNDR depends on:

- Whether FEAT_RNG is implemented
- Whether FEAT_RNG_TRAP is implemented
  - When FEAT_RNG_TRAP is implemented, the value of SCR_EL3.TRNDR

As shown here:

FEAT_RNG	FEAT_RNG_TRAP	SCR_EL3.TRNDR	Value returned by ID_AA64ISAR0_EL1.RNDR
N	N	-	b0000
N	Y	0	b0000
N	Y	1	b0001
Y	x	x	b0001

I_{QSXLV} When FEAT_RNG is not implemented, SCR_EL3.TRNDR can cause the value returned by reads of ID_AA64ISAR0_EL1.RNDR to change. Arm strongly recommends that SCR_EL3.TRNDR is initialized before entering Exception levels below EL3 and not subsequently changed.

15.1.7 ID_AA64PFR0_EL1

R_{JKBJC} ID_AA64PFR0_EL1[55:52] is allocated as the Realm Management Extension (RME) field:

- `0b0000` = Realm Management Extension not implemented.
- `0b0001` = RMEv1 is implemented.

All other values are reserved.

FEAT_RME implements the functionality identified by 0b0001.

15.1.8 ID_AA64PFR1_EL1

R_{NWFLN} ID_AA64PFR1_EL1[31:28] is allocated as the RNDR_trap field:

- `0b0000` = Trapping of RNDR and RNDRRS to EL3 is not supported.

- `0b0001` = Trapping of RNDR and RNDRRS to EL3 is supported, SCR_EL3.TRNDR is implemented.

All other values are reserved.

FEAT_RNG_TRAP implements the functionality identified by 0b0001.

15.1.9 HCR_EL2

RZKLMH HCR_EL2[48] is allocated as the GPF field:

This field controls the reporting of Granule protection faults at EL0 and EL1.

Value	Meaning
`0b0`	This control does not cause exceptions to be routed from EL0 and EL1 to EL2.
`0b1`	Instruction Aborts and Data Aborts due to GPFs from EL0 and EL1 are routed to EL2.

This field resets to an architecturally UNKNOWN value.

This field is RES0 in PEs that do not implement FEAT_RME.

See also:

- 3.4.1 *Exceptions from GPC faults*
- 3.4.3 *Data and Instruction Abort exceptions*

15.1.10 HPFAR_EL2

RZZNQR The description of the HPFAR_EL2.NS is amended to include the case where aborts are taken to Realm EL2 as follows:

Bits [63] NS

Faulting IPA address space.

NS	Meaning
0	Faulting IPA is from the Secure IPA space.
1	Faulting IPA is from the Non-secure IPA space.

For Data Aborts or Instruction Aborts that are taken to Non-secure EL2, this field is RES0, and the address is from the Non-secure IPA space. For Data Aborts or Instruction Aborts that are taken to Realm EL2, this field is RES0, and the address is from the Realm IPA space.

This field resets to an architecturally UNKNOWN value.

See also:

- 5.1.2 *Realm translation regimes*

15.1.11 LORC_EL1, LOREA_EL1, LORN_EL1 and LORSA_EL1

I_{GKXBL} RME makes no changes to the LORC_EL1, LOREA_EL1, LORN_EL1, and LORSA_EL1 registers. These registers are accessible when SCR_EL3.NS == 1.

This means that the LORegion registers are accessible in Non-secure state, Realm state and at EL3. However, LORegions can only be defined in the Non-secure physical address space, regardless of the current Security state.

15.1.12 MDCCSR_EL0

I_{CVBMG} RME introduces changes to EDSCR. MDCCSR_EL0 bits [30:29] are architecturally mapped to External register EDSCR[30:29]. RME introduces changes to EDSCR, but not in the region which is architecturally mapped to MDCCSR_EL0. See 15.3.8 *EDSCR*.

15.1.13 MDCR_EL3

R_{LXYHR} MDCR_EL3[26] is allocated as NSTBE.

Together with the NSTB field, this field controls the owning translation regime and accesses to Trace Buffer control registers from EL2 and EL1.

NSTBE	NSTB	Description
0	00	Trace Buffer owning Security state is Secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
0	01	Trace Buffer owning Security state is Secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 in Non-secure and Realm state generate Trap exceptions to EL3.
0	10	Trace Buffer owning Security state is Non-secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
0	11	Trace Buffer owning Security state is Non-secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 in Secure and Realm state generate Trap exceptions to EL3.
1	10	Trace Buffer owning Security state is Realm state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Secure state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
1	11	Trace Buffer owning security state is Realm state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Secure state. Accesses to Trace Buffer control registers at EL2 and EL1 in Non-secure and Secure state generate Trap exceptions to EL3.

All other encodings are reserved.

This field resets to an architecturally UNKNOWN value.

When FEAT_RME is not implemented, this field is RES0.

I~PKQXC~ There is no encoding for Root state, as self-hosted trace is always prohibited at EL3 (when EL3 uses AArch64).

R~CYJQB~ MDCR_EL3[11] is allocated as NSPBE.

Together with the NSPB field, this field controls the owning translation regime and accesses to Statistical Profiling and Profiling Buffer control registers.

NSPBE	NSPB	Description
0	00	Profiling Buffer uses Secure virtual addresses. Statistical Profiling enabled in Secure state, disabled in Non-secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in all Security states generate Trap exceptions to EL3.
0	01	Profiling Buffer uses Secure virtual addresses. Statistical Profiling enabled in Secure state, disabled in Non-secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure and Realm state generate Trap exceptions to EL3.
0	10	Profiling Buffer uses Non-secure virtual addresses. Statistical Profiling enabled in Non-secure state, disabled in Secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in all Security states generate Trap exceptions to EL3.
0	11	Profiling Buffer uses Non-secure virtual addresses. Statistical Profiling enabled in Non-secure state, disabled in Secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Secure and Realm state generate Trap exceptions to EL3.
1	10	Profiling Buffer uses Realm virtual addresses. Statistical Profiling enabled in Realm state, disabled in Non-secure and Secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in all Security states generate Trap exceptions to EL3.
1	11	Profiling Buffer uses Realm virtual addresses. Statistical Profiling enabled in Realm state, disabled in Non-secure and Secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure and Secure state generate Trap exceptions to EL3.

This field resets to an architecturally UNKNOWN value.

When FEAT_RME is not implemented, this field is RES0.

I~SCCJW~ There is no encoding for Root state, as profiling is always disabled at EL3.

R~KNRDY~ The following additional field is defined:

Bit [4] EDADE, External Debug Access Disable Extended.

Together with MDCR_EL3.EDAD, controls access to breakpoint registers, watchpoint registers and OSLAR_EL1 by an external debugger.

EDADE	EDAD	Meaning
0	0	Access to debug registers by an external debugger is permitted.
0	1	Root and Secure access to debug registers by an external debugger is permitted. Realm and Non-secure access to debug registers by an external debugger is not permitted.
1	0	Root and Realm access to debug registers by an external debugger is permitted. Secure and Non-secure access to debug registers by an external debugger is not permitted.
1	1	Root access to debug registers by an external debugger is permitted. Secure, Non-secure and Realm access to debug registers by an external debugger is not permitted.

When FEAT_RME is not implemented, this bit is RES0.

On a Warm reset, this field resets to 0.

R_{TMHDW} The following additional field is defined:

Bit [3] ETADE, External Trace Access Disable Extended.

Together with MDCR_EL3.ETAD, controls access to PE Trace Unit registers by an external debugger.

ETADE	ETAD	Meaning
0	0	Access to PE Trace Unit registers by an external debugger is permitted.
0	1	Root and Secure access to PE Trace Unit registers by an external debugger is permitted. Realm and Non-secure access to PE Trace Unit registers by an external debugger is not permitted.
1	0	Root and Realm access to PE Trace Unit registers by an external debugger is permitted. Secure and Non-secure access to PE Trace Unit registers by an external debugger is not permitted.
1	1	Root access to PE Trace Unit registers by an external debugger is permitted. Secure, Non-secure and Realm access to PE Trace Unit registers by an external debugger is not permitted.

When FEAT_RME is not implemented, this bit is RES0.

On a Warm reset, this field resets to 0.

R_{CQNJG} The following additional field is defined:

Bit [2] EPMADE, External Performance Monitors Access Disable Extended.

Together with MDCR_EL3.EPMAD, controls access to Performance Monitor registers by an external debugger.

EPMADE	EPMAD	Meaning
0	0	Access to Performance Monitor registers by an external debugger is permitted.

EPMADE	EPMAD	Meaning
0	1	Root and Secure access to Performance Monitor registers by an external debugger is permitted. Realm and Non-secure access to Performance Monitor registers by an external debugger is not permitted.
1	0	Root and Realm access to Performance Monitor registers by an external debugger is permitted. Secure and Non-secure access to Performance Monitor registers by an external debugger is not permitted.
1	1	Root access to Performance Monitor registers by an external debugger is permitted. Secure, Non-secure and Realm access to Performance Monitor registers by an external debugger is not permitted.

If the Performance Monitors Extension does not support external debug interface accesses, this bit is RES0.

When FEAT_RME is not implemented, this bit is RES0.

On a Warm reset, this field resets to 0.

I_PGCQH MDCR_EL3.EDAD/EDADE control which accesses to external Debug registers are permitted.

MDCR_EL3.EPMAD/EPMADE control which accesses to external PMU registers are permitted.

MDCR_EL3.ETAD/ETADE control which accesses to external trace registers are permitted.

Permitting Non-secure accesses or Secure accesses while in Realm state exposes some of the Realm state context. Similarly, permitting Non-secure accesses or Realm accesses while in Secure state exposes some of the Secure state context. These controls allow EL3 software to limit visibility of external registers only to accesses which match the current Security state, and Root accesses.

Root accesses are always permitted, as an entity that can generate Root accesses must be considered trusted.

R_YSJXN The following additional field is defined:

Bit [0] RLTE, Realm Trace enable. Enables tracing in Realm state.

Value	Meaning
0b0	Trace prohibited in Realm state unless overridden by the IMPLEMENTATION DEFINED authentication interface.
0b1	Trace in Realm state is not affected by this bit.

This bit also controls the level of authentication that is required by an external debugger to enable external tracing.

If FEAT_TRF is not implemented, this bit is RES0.

On a Warm reset, this field resets to 0.

Otherwise, RES0.

See also:

- 10.1 *Self-hosted debug*
- 10.2 *Self-hosted trace*
- 11.2 *Required debug authentication*

15.1.14 MFAR_EL3

Holds the faulting PA for Granule Protection Check exceptions taken to EL3.

Access: EL3 only. UNDEFINED for lower Exception levels.

Purpose: Reports faulting physical address for GPC exceptions

Configuration: This register is present only when FEAT_RME is implemented. Otherwise, direct accesses to MFAR_EL3 are UNDEFINED.

Encoding: Allocated encoding

op0	op1	CRn	CRm	op2
0b11	0b110	0b0110	0b0000	0b101

Bit [63] NS

Together with the NSE field, this field reports the physical address space of the access that triggered the Granule Protection Check exception.

NSE	NS	Meaning
0	0	Secure
0	1	Non-secure
1	0	Root
1	1	Realm

This field resets to an architecturally UNKNOWN value.

Bit [62] NSE

Together with the NS field, reports physical address space of the access that triggered the Granule Protection Check exception.

This field resets to an architecturally UNKNOWN value.

Bits [61:52] RES0

Bits [51:48] FPA[51:48]

When FEAT_LPA is implemented, extension to FPA[47:12]. This field resets to an architecturally UNKNOWN value.

When FEAT_LPA is not implemented, this field is RES0.

Bits [47:12] FPA[47:12]

Bits [47:12] of the faulting physical address.

For implementations with fewer than 48 physical address bits, the corresponding upper bits in this field are RES0.

This field resets to an architecturally UNKNOWN value.

Bits [11:0] RES0

I_{YLYQM} This register holds the input PA for the Granule Protection Check that triggered the taken exception. For Granule Protection Check exceptions on a stage 1 or 2 translation table walk, this is the address of the descriptor.

15.1.15 OSECCR_EL1

See 15.3.5 *EDECCR*.

15.1.16 PAR_EL1

R_{CGLQG} When PAR_EL1.F == 0b0, PAR_EL1[11] is allocated as the NSE field.

Reports the NSE attribute for a translation table entry from the EL3 translation regime.

For a result from a Secure, Non-secure, or Realm translation regime, this bit is UNKNOWN.

R_{QXRWL} When PAR_EL1.F == 0b0, PAR_EL1.NS reports the NS attribute for a translation table entry from an EL3, Secure, or Realm translation regime.

For a result from a S1E1 or S1E0 operation on the Realm EL1&0 translation regime, this bit is UNKNOWN.

For a result from a Non-secure regime, this bit is UNKNOWN.

I_{YGMTF} In Realm state, the EL1&0 translation regime does not have an NS bit. The behavior of Realm EL1 and EL0 AT instructions mirrors the treatment for Non-secure translation regimes.

I_{GFHZM} The behavior of PAR_EL1.NS when PAR_EL1.F == 0b0 for results from Non-secure translation regimes is unchanged.

R_{GGQJP} When PAR_EL1.F == 0b1, the following additional PAR_EL1.FST are defined:

xFSC	Meaning
`0b100011`	Granule protection fault on translation table walk or hardware update of translation table, level -1.
`0b1001xx`	Granule protection fault on translation table walk or hardware update of translation table, level *xx*.

The encoding 100011 is only present if FEAT_LPA2 is implemented.

15.1.17 PMBIDR_EL1

R_{ZFDGT} The description of PMBIDR_EL1.P is modified:

The value read from this field depends on the current Exception level and the Effective values of MDCR_EL3.NSPB, MDCR_EL3.NSPBE, and MDCR_EL2.E2PB:

- If EL3 is implemented, and the owning Security state is Secure state, this bit reads as one from:
  - Non-secure EL1 and Non-secure EL2.
  - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
  - If Secure EL2 is implemented and enabled, and MDCR_EL2.E2PB is `0b00`, Secure EL1.
- If EL3 is implemented, and the owning Security state is Non-secure state, this bit reads as one from:
  - Secure EL1.
  - If Secure EL2 is implemented, Secure EL2.
  - If EL2 is implemented and MDCR_EL2.E2PB is `0b00`, Non-secure EL1.
  - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
- If FEAT_RME is implemented, and the owning Security state is Realm state, this bit reads as one from:
  - Non-secure EL1 and Non-secure EL2.

- Secure EL1 and Secure EL2.
- If MDCR_EL2.E2PB is 0b00, Realm EL1.

- If EL3 is not implemented, EL2 is implemented, and MDCR_EL2.E2PB is `0b00`, this bit reads as one from EL1.
- Otherwise, this bit reads as zero.

I_{JJQHP} The changes to the description cover accesses from Realm state to PMBIDR_EL1.P and accesses from other Security states when the buffer is owned by Realm state.

15.1.18 PMBSR_EL1

R_{MNQJL} PMBSR_EL1.EC == `0b01_1110` is allocated as *Granule Protection Check fault, other than GPF, on write to profiling buffer*. PMBSR_EL1.MSS is RES0 for this exception class.

I_{HJKFC} PMBSR_EL1.EC == `0b01_1110` matches the ESR_ELx.EC value used for GPC exceptions.

R_{RHCZS} In the MSS encoding *MSS encoding for stage 1 or stage 2 Data Aborts on write to buffer* the following additional FSC values are added:

FSC	Meaning
`0b100011`	Granule protection fault on translation table walk or hardware update of translation table, level -1.
`0b1001xx`	Granule protection fault on translation table walk or hardware update of translation table, level *xx*.
`0b101000`	Granule protection fault not on translation table walk or hardware update of translation table.

The encoding 100011 is only present if FEAT_LPA2 is implemented.

15.1.19 PMCCFILTR_EL0

R_{JYKYK} The following is added to the description of PMCCFILTR_EL0.P:

If FEAT_RME is implemented, then counting in Realm EL1 is further controlled by the PMCCFILTR_EL0.RLK bit.

R_{MWRFB} The following additional field is defined:

Bit [22] RLK, Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Realm EL1 are counted. Otherwise, events in Realm EL1 are not counted.

On a Warm reset, this field resets to an architecturally UNKNOWN value.

If FEAT_RME is not implemented, this field is RES0.

R_{DXYZN} The following is added to the description of PMCCFILTR_EL0.U:

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMCCFILTR_EL0.RLU bit.

R_{BHKHM} The following additional field is defined:

Bit [21] RLU, Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Realm EL0 are counted. Otherwise, events in Realm EL0 are not counted.

On a Warm reset, this field resets to an architecturally UNKNOWN value.

If FEAT_RME is not implemented, this field is RES0.

I_{LMMBW} The RLU field is also added in the mapped AArch32 System register.

R_{MSFWT} The following is added to the description of PMCCFILTR_EL0.NSH:

If FEAT_RME is implemented, then counting in Realm EL2 is further controlled by the PMCCFILTR_EL0.RLH bit.

R_{JDMVX} The following additional field is defined:

Bit [20] RLH, Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Realm EL2 are counted. Otherwise, events in Realm EL2 are not counted.

On a Warm reset, this field resets to an architecturally UNKNOWN value.

If FEAT_RME is not implemented, this field is RES0.

I_{QYDGX} These controls give equivalent functionality to what is available for Non-secure and Secure states. The order of the fields controlling counting in Realm state mirrors that of existing fields.

I_{SLJNJ} Where the current description of PMCCFILTR_EL0.M refers to “Secure EL3”, in a system implementing FEAT_RME these references should be “EL3”. However, the function of PMCCFILTR_EL0.M is unchanged by FEAT_RME.

15.1.20 PMEVTYPER<n>_EL0

R_{TPBKX} The following is added to the description of PMEVTYPER<n>_EL0.P:

If FEAT_RME is implemented, then counting in Realm EL1 is further controlled by the PMEVTYPER<n>_EL0.RLK bit.

R_{CTLCH} The following additional field is defined:

Bit [22] RLK, Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted. Otherwise, events in Realm EL1 are not counted.

On a Warm reset, this field resets to an architecturally UNKNOWN value.

If FEAT_RME is not implemented, this field is RES0.

R_{GXFDT} The following is added to the description of PMEVTYPER<n>_EL0.U:

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMEVTYPER<n>_EL0.RLU bit.

R_{JHHWN} The following additional field is defined:

Bit [21] RLU, Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted. Otherwise, events in Realm EL0 are not counted.

On a Warm reset, this field resets to an architecturally UNKNOWN value.

If FEAT_RME is not implemented, this field is RES0.

I_{XGWGY} The RLU field is also added in the mapped AArch32 System register.

R_{KZYPH} The following is added to the description of PMEVTYPER<n>_EL0.NSH:

If FEAT_RME is implemented, then counting in Realm EL2 is further controlled by the PMEVTYPER<n>_EL0.RLH bit.

R_XMHZL The following additional field is defined:

Bit [20] RLH, Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted. Otherwise, events in Realm EL2 are not counted.

On a Warm reset, this field resets to an architecturally UNKNOWN value.

If FEAT_RME is not implemented, this field is RES0.

I_DVYYG Where the current description of PMEVTYPER<n>_EL0.M refers to "Secure EL3", in a system implementing FEAT_RME these references should be "EL3". However, the function of PMEVTYPER<n>_EL0.M is unchanged by FEAT_RME.

15.1.21 RNDR and RNDRRS

I_SQMSF The behavior of direct reads of RNDR or RNDRRS depends on whether FEAT_RNG and FEAT_RNG_TRAP are implemented:

ID_AA64PFR1_EL1.RNDR_trap	ID_AA64ISAR0_EL1.RNDR	SCR_EL3.TRNDR	Behavior of direct reads
`0b0000`	`0b0000`	-	UNDEFINED
`0b0000`	`0b0001`	-	Reads are not trapped to EL3
`0b0001`	`0b0000`	`0b0`	UNDEFINED
`0b0001`	x	`0b1`	Reads trap to EL3
`0b0001`	`0b0001`	`0b0`	Reads are not trapped to EL3

15.1.22 TRBIDR_EL1

R_SJCFN The description of TRBIDR_EL1.P is modified:

The value read from this field depends on the current Exception level and the Effective values of MDCR_EL3.NSTB, MDCR_EL3.NSTBE, and MDCR_EL2.E2TB:

- If EL3 is implemented, and the owning Security state is Secure state, this bit reads as one from:
  - Non-secure EL1 and Non-secure EL2.
  - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
  - If Secure EL2 is implemented and enabled, and MDCR_EL2.E2TB is `0b00`, Secure EL1.
- If EL3 is implemented, and the owning Security state is Non-secure state, this bit reads as one from:
  - Secure EL1.
  - If Secure EL2 is implemented, Secure EL2.
  - If EL2 is implemented and MDCR_EL2.E2TB is `0b00`, Non-secure EL1.
  - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
- If FEAT_RME is implemented, and the owning Security state is Realm state, this bit reads as one from:
  - Non-secure EL1 and Non-secure EL2.
  - Secure EL1 and Secure EL2.
  - If MDCR_EL2.E2TB is 0b00, Realm EL1.
- If EL3 is not implemented, EL2 is implemented, and MDCR_EL2.E2TB is `0b00`, this bit reads as one from EL1.
- Otherwise, this bit reads as zero.

I_{QRLYK} The changes to the description cover accesses from Realm state to TRBIDR_EL1.P and accesses from other Security states when the buffer is owned by Realm state.

15.1.23 TRBSR_EL1

R_{WBJLC} TRBSR_EL1.EC == `0b01_1110` is allocated as *Granule Protection Check fault, other than GPF, on write to trace buffer*. TRBSR_EL1.MSS is RES0 for this exception class.

R_{GYDLQ} In the MSS encoding *MSS encoding for stage 1 or stage 2 Data Aborts on write to buffer* the following additional FSC values are added:

FSC	Meaning
`0b100011`	Granule protection fault on translation table walk or hardware update of translation table, level -1.
`0b1001xx`	Granule protection fault on translation table walk or hardware update of translation table, level *xx*.
`0b101000`	Granule protection fault not on translation table walk or hardware update of translation table.

The encoding 100011 is only present if FEAT_LPA2 is implemented.

15.1.24 TRCAUTHSTATUS

See 15.3.14 *TRCAUTHSTATUS*.

15.1.25 TRCDEVARCH

See 15.3.15 *TRCDEVARCH*.

15.1.26 TRCIDR6

See 15.3.16 *TRCIDR6*.

15.1.27 GPCCR_EL3, Granule Protection Check Control Register

Access: EL3 only. UNDEFINED for lower Exception levels.

Purpose: Control register for Granule Protection Checks

Configuration: This register is present only when FEAT_RME is implemented. Otherwise, direct accesses to GPCCR_EL3 are UNDEFINED.

Encoding: Allocated encoding

op0	op1	CRn	CRm	op2
`0b11`	`0b110`	`0b0010`	`0b0001`	`0b110`

Bits [23:20] L0GPTSZ

Level 0 GPT entry size.

This field advertises the number of least-significant address bits protected by each entry in the level 0 GPT.

Value	Meaning
`0b0000`	30-bits. Each entry covers 1GB of address space.
`0b0100`	34-bits. Each entry covers 16GB of address space.
`0b0110`	36-bits. Each entry covers 64GB of address space.
`0b1001`	39-bits. Each entry covers 512GB of address space.

This field is read-only.

See also:

- 4.5.5 *Lookup process*

Bit [17] GPCP

Granule Protection Check Priority

This control governs behavior of granule protection checks on fetches of stage 2 Table descriptors.

Value	Meaning
`0b0`	All GPC faults are reported with a priority consistent with the GPC being performed on any access to physical address space.
`0b1`	A GPC fault for the fetch of a Table descriptor for a stage 2 translation table walk might not be generated or reported. All other GPC faults are reported with a priority consistent with the GPC being performed on any access to physical address space.

This bit resets to an architecturally UNKNOWN value.

This bit is permitted to be cached in a TLB.

See also:

- 3.4.1 *Exceptions from GPC faults*
- 4.5.1 *GPC behavior overview*

Bit [16] GPC

Granule Protection Check Enable

Value	Meaning
`0b0`	Granule protection checks are disabled. Accesses are not prevented by this mechanism.
`0b1`	All accesses to physical address spaces are subject to granule protection checks, except for fetches of GPT information and accesses governed by the GPCCR_EL3.GPCP control.

This bit resets to 0.

This bit is permitted to be cached in a TLB if any stage of translation is enabled.

Bits [15:14] PGS

Physical Granule size

Value	Meaning
`0b00`	4KB
`0b01`	64KB
`0b10`	16KB

Other values are reserved.

Granule sizes not supported for stage 1 and not supported for stage 2, as advertised in ID_AA64MMFR0_EL1, are reserved.

For example, if ID_AA64MMFR0_EL1.TGran16 == `0b0000` and ID_AA64MMFR0_EL1.TGran16_2 == `0b0001` then the PGS encoding `0b10` is reserved.

The value of this field is permitted to be cached in a TLB.

This field resets to an architecturally UNKNOWN value.

Bits [13:12] SH

GPT fetch Shareability attribute

Value	Meaning
`0b00`	Non-shareable
`0b10`	Outer Shareable
`0b11`	Inner Shareable

Other values are reserved.

Fetches of GPT information are made with the Shareability attribute configured in this field.

If both ORGN and IRGN are configured with Non-cacheable attributes, it is invalid to configure this field to any value other than `0b10`.

This field resets to an architecturally UNKNOWN value.

Bits [11:10] ORGN

GPT fetch Outer cacheability attribute

Value	Meaning
`0b00`	Normal memory, Outer Non-cacheable.
`0b01`	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.
`0b10`	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.
`0b11`	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.

Fetches of GPT information are made with the Outer cacheability attributes configured in this field.

This field resets to an architecturally UNKNOWN value.

Bits [9:8] IRGN

GPT fetch Inner cacheability attribute

Value	Meaning
`0b00`	Normal memory, Inner Non-cacheable.
`0b01`	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable.
`0b10`	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable.
`0b11`	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.

Fetches of GPT information are made with the Inner cacheability attributes configured in this field.

This field resets to an architecturally UNKNOWN value.

Bits [2:0] PPS

Protected Physical Address Size

The bit width of the memory region protected by GPTBR_EL3.

Value	Meaning	Usable address space
`0b000`	32 bits	4GB
`0b001`	36 bits	64GB
`0b010`	40 bits	1TB
`0b011`	42 bits	4TB
`0b100`	44 bits	16TB
`0b101`	48 bits	256TB
`0b110`	52 bits	4PB

Other values are reserved.

Configuration of this field to a value exceeding the implemented physical address size is invalid.

The value of this field is permitted to be cached in a TLB.

This field resets to an architecturally UNKNOWN value.

See also:

- *3.4.2 Granule protection check exceptions*
- *4.5 Granule Protection Checks*

15.1.28 GPTBR_EL3, Granule Protection Table Base Register

Access: EL3 only. UNDEFINED for lower Exception levels.

Purpose: Control register for Granule Protection Table base address

Configuration: This register is present only when FEAT_RME is implemented. Otherwise, direct accesses to GPTBR_EL3 are UNDEFINED.

Encoding: Allocated encoding

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0001	0b100

This register resets to an architecturally UNKNOWN value.

Bits [39:0] BADDR

Base Address for the level 0 GPT.

This field represents bits [51:12] of the level 0 GPT base address.

The level 0 GPT is aligned in memory to the greater of:

- the size of the level 0 GPT in bytes.
- 4KB.

Bits [*x*:0] of the base address are treated as zero, where:

- x = Max(*pps* - *l0gptsz* + 2, 11)
- *pps* is derived from GPCCR_EL3.PPS as follows:

PPS	*pps*
0b000	32
0b001	36
0b010	40
0b011	42
0b100	44
0b101	48
0b110	52

- *l0gptsz* is derived from GPCCR_EL3.L0GPTSZ as follows:

L0GPTSZ	*l0gptsz*
0b0000	30
0b0100	34
0b0110	36
0b1001	39

If x is greater than 11, then BADDR[x - 12:0] are RES0.

See also:

- 4.5 *Granule Protection Checks*

15.1.29 SCR_EL3

Bit [62] NSE

This field, evaluated with SCR_EL3.NS, selects the Security state of EL2 and lower Exception levels.

This field resets to an architecturally UNKNOWN value.

This field is RES0 in PEs that do not implement FEAT_RME.

Bit [48] GPF

This field controls the reporting of Granule protection faults at EL0, EL1 and EL2.

Value	Meaning
`0b0`	This control does not cause exceptions to be routed from EL0, EL1 or EL2 to EL3.
`0b1`	GPFs at EL0, EL1 and EL2 are routed to EL3 and reported as Granule Protection Check exceptions.

This field resets to an architecturally UNKNOWN value.

This field is RES0 in PEs that do not implement FEAT_RME.

Bit [40] TRNDR

This field controls the trapping of RNDR and RNDRRS instructions.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped and has no affect on reads of ID_AA64ISAR0_EL1.RNDR.
`0b1`	Reads of RNDR and RNDRRS are trapped to EL3. When FEAT_RNG is not implemented, reads of ID_AA64ISAR0_EL1.RNDR return the value b0001.

This field resets to 0.

This field is RES0 in PEs that do not implement FEAT_RNG_TRAP.

See also:

- *3.3 Security states*
- *3.4 Exceptions*

15.1.30 TRCACATR<n>

See also:

- *15.3.13 TRCACATR<n>*

15.1.31 TRCVICTLR

See also:

- *15.3.17 TRCVICTLR*

15.2 GIC registers

I_{RFDPL} Behavior of GIC registers is described in the GIC specification [7].

R_{TWGCB} Accesses to ICC, ICH and ICV registers from Realm state are treated in the same way as accesses from Non-secure state.

R_{CVKXN} Accesses to ICC and ICH registers from Root state are treated in the same way as accesses from Secure state.

R_{YCXDF} Accesses to GICD, GICR and GITS registers using the Realm physical address space are treated the same as accesses via the Non-secure physical address space.

R_{NYNDC} Accesses to GICD, GICR and GITS registers using the Root physical address space are treated the same as accesses via the Secure physical address space.

I_{RTBBF} For accesses to memory-mapped GIC registers, the rules describe the behavior of accesses arriving at the GIC. Such accesses would also be subject to granule protection checks on the PE. Arm expects the granule protection table configuration corresponding to GIC register frames to be configured as *All Accesses Permitted*.

15.2.1 ICC_CTLR_EL3

R_{RBBSN} A PE that implements FEAT_RME reports ICC_CTLR_EL3.nDS==1.

I_{PHJGR} This indicates that the PE does not support the disabling of security within the GIC.

15.2.2 ICC_SRE_ELx

I_{SNNRS} A PE that implements FEAT_RME does not support legacy operation.

15.2.3 ICH_VTR_EL2

R_{XHDDR} A PE that implements FEAT_RME reports ICH_VTR_EL2.DVIM==1.

I_{LNTFT} This indicates that the PE supports masking of directly-injected virtual interrupts from the GIC IRI.

15.3 External registers

15.3.1 CNTReadBase and CNTControlBase (Memory-mapped counter module)

R_{CBZWL} In a system that supports the Realm Management Extension, CNTControlBase is accessible only by Root accesses.

I_{ZFVGL} CNTReadBase is accessible in all physical address spaces.

See also:

- *Counter module control and status register summary*, Armv8-A [2].

15.3.2 CNTBaseN, CNTEL0BaseN, and CNTCTLBase (Memory-mapped timer components)

R_{LTXDR} For any register in CNTBaseN, CNTEL0BaseN, or CNTCTLBase described in Armv8-A [2] as permitting Non-secure access, it is IMPLEMENTATION DEFINED whether Root and Realm accesses are permitted. If not permitted, the register behaves as RES0 for Root and Realm accesses.

R_{LJZFN} For any register in CNTBaseN, CNTEL0BaseN, or CNTCTLBase described in Armv8-A [2] as only permitting Secure accesses:

- For Root accesses, it is IMPLEMENTATION DEFINED whether accesses are permitted or behave as RES0.
- For Realm accesses, the register behaves as RES0.

I_{CMJWN} Where hardware does not permit Realm accesses to Non-secure timers, software in Realm state can still access the timers by mapping them into the VA or IPA space as Non-secure. Similarly, software at EL3 can map Secure and Non-secure timers into its VA space with appropriate TTD attributes. Arm recommends that EL3 software and Realm EL2 software always maps timers with TTD attributes for the owning Security state of the timer.

I_{NXPVW} CNTNSAR is not extended to allow allocating of a timer to Realm or Root state.

I_{LCMVV} Section *Providing a complete set of features in a system level implementation* in Armv8-A [2] gives an example memory-mapped Generic Timer implementation. In that example, Frame 3 is described as the *Secure EL3 timer*. In an RME system which used the example implementation, this timer would be accessible from Secure state, as well as from EL3. Therefore, EL3 software using the timer would not be protected from interference from software in Secure state.

For the system-register mapped, the EL3 physical timer can be protected from secure accesses by setting SCR_EL3.ST==0.

See also:

- *Memory-mapped timer components*, Armv8-A [2].

15.3.3 CTIAUTHSTATUS

R_{KCHLM} The following additional field is defined:

Bit [27:24] Reserved, RAZ.

R_{QYNLQ} The following additional field is defined:

Bit [15:12] Reserved, RAZ.

15.3.4 DBGAUTHSTATUS_EL1

R_{GMCVY} The following additional field is defined:

Bit [27:26] RTNID, Root non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RTID.

All other values are reserved.

R_{MSKFV} The following additional field is defined:

Bit [25:24] RTID, Root invasive debug.

Value	Meaning
0b00	Not implemented.
0b10	Implemented and disabled. ExternalRootInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalRootInvasiveDebugEnabled() == TRUE.

All other values are reserved.

R_{MWLLM} The following additional field is defined:

Bit [15:14] RLNID, Realm non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RLID.

R_{XDYTD} The following additional field is defined:

Bit [13:12] RLID, Realm invasive debug.

Value	Meaning
0b00	Not implemented.
0b10	Implemented and disabled. ExternalRealmInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalRealmInvasiveDebugEnabled() == TRUE.

All other values are reserved.

See also:

- *11.2 Required debug authentication*

15.3.5 EDECCR

I_{KSYGK} When OSLSR_EL1.OSLK==1, OSECCR_EL1[31:0] is architecturally mapped to EDECCR[31:0]. Changes described here for EDECCR also apply to OSECCR_EL1.

R_{XXKSG} The following additional field is defined:

Bits [18:16] RLE<n>, Coarse-grained Realm exception catch for EL<n>

RLE<n>	RLR<n>	Meaning
0	0	Exception Catch debug events are disabled for Realm Exception level <n>.
0	1	Exception Catch debug events are enabled for exception returns to Realm Exception level <n>.

RLE<n>	RLR<n>	Meaning
1	0	Exception Catch debug events are enabled for exception entry and exception return to Realm Exception level <n>.
1	1	Exception Catch debug events are enabled for exception entry to Realm Exception level <n>.

RLE0 is RES0.

The following resets apply:

- On a Cold reset, this field resets to 0
- On an External debug reset, the value of this field is unchanged.
- On a Warm reset, the value of this field is unchanged.

This field is RES0 when FEAT_RME is not implemented.

R_ZQZBT The following additional field is defined:

Bits [22:20] RLR<n>, Coarse-grained Realm exception catch for EL<n>

Controls Realm exception catch on exception return to EL<n> in conjunction with RLE<n>.

The following resets apply:

- On a Cold reset, this field resets to 0
- On an External debug reset, the value of this field is unchanged.
- On a Warm reset, the value of this field is unchanged.

This field is RES0 when FEAT_RME is not implemented.

I_QRSKX Exception catch on exception entry to EL3 is controlled by EDECCR.SE3. Exception catch on exception return to EL3 is controlled by EDECCR.SR3. This is unchanged by the introduction of FEAT_RME.

15.3.6 EDPRCR

R_HPCYG All writes to CWRR are ignored.

I_RBBQG This bit allowed a debugger to request a warm reset. It was in deprecated pre-FEAT_RME, with a recommendation that writes be ignored.

15.3.7 EDPRSR

R_KYQKK The following additional field is defined:

Bit [14] EDADE, External Debug Access Disable Extended status.

Together with EDPRSR.EDAD, reports whether access to breakpoint registers, watchpoint registers and OSLAR_EL1 by an external debugger is permitted.

EDADE	EDAD	Meaning
0	0	Access to debug registers by an external debugger is permitted.
0	1	Root and Secure access to debug registers by an external debugger is permitted. Realm and Non-secure access to debug registers by an external debugger is not permitted.
1	0	Root and Realm access to debug registers by an external debugger is permitted.

EDADE	EDAD	Meaning
		Secure and Non-secure access to debug registers by an external debugger is not permitted.
1	1	Root access to debug registers by an external debugger is permitted.
		Secure, Non-secure and Realm access to debug registers by an external debugger is not permitted.

When FEAT_RME is not implemented, this bit is RES0.

R_GGHGW The following additional field is defined:

Bit [15] ETADE, External Trace Access Disable Extended status.

Together with EDPRSR.ETAD, reports whether access to PE Trace Unit registers by an external debugger is permitted.

ETADE	ETAD	Meaning
0	0	Access to PE Trace Unit registers by an external debugger is permitted.
0	1	Root and Secure access to PE Trace Unit registers by an external debugger is permitted.
		Realm and Non-secure access to PE Trace Unit registers by an external debugger is not permitted.
1	0	Root and Realm access to PE Trace Unit registers by an external debugger is permitted.
		Secure and Non-secure access to PE Trace Unit registers by an external debugger is not permitted.
1	1	Root access to PE Trace Unit registers by an external debugger is permitted.
		Secure, Non-secure and Realm access to PE Trace Unit registers by an external debugger is not permitted.

When FEAT_RME is not implemented, this bit is RES0.

R_NCKYV The following additional field is defined:

Bit [16] EPMADE, Performance Monitor Access Disable Extended status.

Together with EDPRSR.EPMAD, reports whether access to the Performance Monitor registers by an external debugger is permitted.

EPMADE	EPMAD	Meaning
0	0	Access to Performance Monitor registers by an external debugger is permitted.
0	1	Root and Secure access to Performance Monitor registers by an external debugger is permitted.
		Realm and Non-secure access to Performance Monitor registers by an external debugger is not permitted.
1	0	Root and Realm access to Performance Monitor registers by an external debugger is permitted.
		Secure and Non-secure access to Performance Monitor registers by an external debugger is not permitted.
1	1	Root access to Performance Monitor registers by an external debugger is permitted.
		Secure, Non-secure and Realm access to Performance Monitor registers by an external debugger is not permitted.

When FEAT_RME is not implemented, this bit is RES0.

15.3.8 EDSCR

R_{NHZNF} In EDSCR, the description of the TFO field is amended to:

Bit [31] TFO

Trace Filter Override. Overrides the Trace Filter controls allowing the external debugger to trace any visible Exception level.

When FEAT_RME is implemented:

TFO	Meaning
0	Trace Filter controls are not affected.
1	Trace Filter controls in TRFCR_EL1 and TRFCR_EL2 are ignored.
	Trace Filter controls in TRFCR and HTRFCR are ignored.

When OSLSR_EL1.OSLK == 1, this bit can be indirectly read and written through the MDSCR_EL1 and DBGDSCRext System registers.

This bit is ignored by the PE when any of the following is true:

- `ExternalSecureNoninvasiveDebugEnabled()` == FALSE and the Effective value of MDCR_EL3.STE == 1.
- FEAT_RME is implemented, `ExternalRealmNoninvasiveDebugEnabled()` == FALSE and the Effective value of MDCR_EL3.RLTE == 1.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

R_{ZKZBY} In EDSCR, the description of the INTdis field is amended to:

Bit [23:22] INTdis

Interrupt disable. Disables taking interrupts in Non-debug state.

When FEAT_RME is implemented:

INTdis	Meaning
00	Masking of interrupts is controlled by PSTATE and interrupt routing controls.
01	If `ExternalInvasiveDebugEnabled()` == TRUE, then all interrupts taken to Non-secure state are masked.
	If `ExternalSecureInvasiveDebugEnabled()` == TRUE, then all interrupts taken to Secure state are masked.
	If `ExternalRealmInvasiveDebugEnabled()` == TRUE, then all interrupts taken to Realm state are masked.
	If `ExternalRootInvasiveDebugEnabled()` == TRUE, then all interrupts taken to Root state are masked.

All interrupts includes virtual and SError interrupts.

Bit[23] of this register is RES0.

When OSLSR_EL1.OSLK == 1, this field can be indirectly read and written through the MDSCR_EL1 and DBGDSCRext System registers.

This field has no effect when `ExternalInvasiveDebugEnabled()` == FALSE.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

R_{NFRKD} In EDSCR, a new field is added:

Bit [15] NSE

Together with the NS field, this field gives the current Security state.

In Non-debug state, this bit is UNKNOWN.

Access to this field is RO.

This field is RES0 in PEs that do not implement FEAT_RME.

R_{FRMZS} In EDSCR, the description of the NS field is amended to:

Bit [18] NS

Together with the NSE field, gives the current Security state.

R_{NHJXL} In EDSCR, collectively the NS and NSE fields give the current the Security state:

NSE	NS	Meaning
0	0	Secure
0	1	Non-secure
1	0	Root
1	1	Realm

R_{BTQHH} In EDSCR, the description of the SDD field is amended to:

Bit [16] SDD, EL3 debug disabled

Reports the inverse of ExternalRootInvasiveDebugEnabled()

Access to this field is RO.

I_{RRWFY} For PEs that do not implement FEAT_RME, the definition of EDSCR.SDD is unchanged.

See also:

- 11.2 *Required debug authentication*

15.3.9 ERR<n>ADDR

R_{SCPBN} In ERR<n>ADDR, a new field is added:

Bit [59] NSE

Together with the NS field, this field reports the address space of PADDR.

This field is RES0 in PEs that do not implement FEAT_RME.

The following resets apply:

- On an Error recovery reset, the value of this field is unchanged.
- On a Cold reset, this field resets to an architecturally unknown value.

R_{PJTTC} In ERR<n>ADDR, the description of the NS field is amended to:

Bit [63] NS

Together with the NSE field, reports the address space of PADDR.

R_{LMNTR} In ERR<n>ADDR, collectively the NS and NSE fields report the address space of PADDR:

NSE	NS	Meaning
0	0	Secure
0	1	Non-secure
1	0	Root
1	1	Realm

R_{WZPGS} In ERR<n>ADDR, field SI indicates the validity of both NS and NSE as follows:

Bit [62] SI

Address Space Incorrect. Indicates whether ERR<n>ADDR.NS and ERR<n>ADDR.NSE are valid. The possible values of this bit are:

ERR<n>ADDR.SI	Meaning
0	ERR<n>ADDR.NS and ERR<n>ADDR.NSE are correct. That is, it matches the programmers' view of the physical address space of the location recorded in PADDR.
1	ERR<n>ADDR.NS and ERR<n>ADDR.NSE might not be correct and might not match the programmers' view of the physical address space of the location recorded in PADDR.

15.3.10 PMAUTHSTATUS

R_{JJFJN} The following additional field is defined:

Bit [27:26] RTNID, Root non-invasive debug.

This field holds the same value as DBGAUTHSTATUS_EL1.RTNID.

All other values are reserved.

R_{ZMKGY} The following additional field is defined:

Bit [25:24] RTID, Root invasive debug.

Value	Meaning
0b00	Not implemented.

All other values are reserved.

R_{CSZNL} The following additional field is defined:

Bit [15:14] RLNID, Realm non-invasive debug.

This field holds the same value as DBGAUTHSTATUS_EL1.RLNID.

R_{VKKWM} The following additional field is defined:

Bit [13:12] RLID, Realm invasive debug.

Value	Meaning
0b00	Not implemented.

All other values are reserved.

15.3.11 PMEVTYPER<n>_EL0

See 15.1.20 *PMEVTYPER<n>_EL0*.

15.3.12 PMPCSR

RJVGXB In PMPCSR, a new field is added:

Bit [59] NSE

Together with the NS field, indicates the Security state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

This field is RES0 in PEs that do not implement FEAT_RME.

RTNYDH In PMPCSR, the description of the NS field is amended to:

Bit [63] NS

Together with the NSE field, indicates the Security state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

RWLBCZ In PMPCSR, collectively the NS and NSE fields indicate the Security state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample:

NSE	NS	Meaning
0	0	Secure
0	1	Non-secure
1	0	Root
1	1	Realm

15.3.13 TRCACATR<n>

RXLQLJ The description of the EXLEVEL_S_EL3 field is changed to:

Bit [11] EXLEVEL_S_EL3

EL3 address comparison control. Controls whether a comparison can occur at EL3.

EXLEVEL_S_EL3	Meaning
0	The Address Comparator performs comparisons in EL3.
1	The Address Comparator does not perform comparisons in EL3.

On a Trace unit reset, this field resets to an architecturally unknown value.

R_{SPHFT} In TRCACATR<n>, the following new fields are added:

Bit [16] EXLEVEL_RL_EL0

Realm EL0 address comparison control. Controls whether a comparison can occur at EL0 in Realm state.

Case	Meaning
EXLEVEL_RL_EL0 == EXLEVEL_NS_EL0	The Address Comparator performs comparisons in Realm EL0.
EXLEVEL_RL_EL0 != EXLEVEL_NS_EL0	The Address Comparator does not perform comparisons in Realm EL0.

On a Trace unit reset, this field resets to an architecturally unknown value.

Bit [17] EXLEVEL_RL_EL1

Realm EL1 address comparison control. Controls whether a comparison can occur at EL1 in Realm state.

Case	Meaning
EXLEVEL_RL_EL1 == EXLEVEL_NS_EL1	The Address Comparator performs comparisons in Realm EL1.
EXLEVEL_RL_EL1 != EXLEVEL_NS_EL1	The Address Comparator does not perform comparisons in Realm EL1.

On a Trace unit reset, this field resets to an architecturally unknown value.

Bit [18] EXLEVEL_RL_EL2

Realm EL2 address comparison control. Controls whether a comparison can occur at EL2 in Realm state.

Case	Meaning
EXLEVEL_RL_EL2 == EXLEVEL_NS_EL2	The Address Comparator performs comparisons in Realm EL2.
EXLEVEL_RL_EL2 != EXLEVEL_NS_EL2	The Address Comparator does not perform comparisons in Realm EL2.

On a Trace unit reset, this field resets to an architecturally unknown value.

15.3.14 TRCAUTHSTATUS

R_{GCLFZ} The following additional field is defined:

Bit [27:26] RTNID, Root non-invasive debug.

This field holds the same value as DBGAUTHSTATUS_EL1.RTNID.

All other values are reserved.

R_{SWNWY} The following additional field is defined:

Bit [25:24] RTID, Root invasive debug.

Value	Meaning
0b00	Not implemented.

All other values are reserved.

R_{CMBXH} The following additional field is defined:

Bit [15:14] RLNID, Realm non-invasive debug.

This field holds the same value as DBGAUTHSTATUS_EL1.RLNID.

R_{SLDWN} The following additional field is defined:

Bit [13:12] RLID, Realm invasive debug.

Value	Meaning
0b00	Not implemented.

All other values are reserved.

15.3.15 TRCDEVARCH

R_{KFVHH} In TRCDEVARCH.REVISION, the following additional value is defined:

REVISON	Meaning
0b0010	ETEv1.2, FEAT_ETEv1p2.

15.3.16 TRCIDR6

R_{CKVYC} In TRCIDR6, the following fields are added:

Bit [0] EXLEVEL_RL_EL0

When FEAT_ETEv1p2 is implemented:

Case	Meaning
0	Realm EL0 is not implemented.
1	Realm EL0 is implemented.

When FEAT_ETEv1p2 is not implemented, this field is RES0.

Bit [1] EXLEVEL_RL_EL1

When FEAT_ETEv1p2 is implemented:

Case	Meaning
0	Realm EL1 is not implemented.
1	Realm EL1 is implemented.

When FEAT_ETEv1p2 is not implemented, this field is RES0.

Bit [2] EXLEVEL_RL_EL2

When FEAT_ETEv1p2 is implemented:

Case	Meaning
0	Realm EL2 is not implemented.
1	Realm EL2 is implemented.

When FEAT_ETEv1p2 is not implemented, this field is RES0.

15.3.17 TRCVICTLR

R_{DXWXM} The description of the EXLEVEL_S_EL3 field is changed to:

Bit [19] EXLEVEL_S_EL3

Filter instruction trace for EL3.

EXLEVEL_S_EL3	Meaning
0	The trace unit generates instruction trace for EL3.
1	The trace unit does not generate instruction trace for EL3.

On a Trace unit reset, this field resets to an architecturally unknown value.

R_{XMCXH} In TRCVICTLR, the following new fields are added:

Bit [24] EXLEVEL_RL_EL0

Filter instruction trace for EL0 in Realm state.

Case	Meaning
EXLEVEL_RL_EL0 == EXLEVEL_NS_EL0	The trace unit generates instruction trace for EL0 in Realm.
EXLEVEL_RL_EL0 != EXLEVEL_NS_EL0	The trace unit does not generate instruction trace for EL0 in Realm state.

On a Trace unit reset, this field resets to an architecturally unknown value.

Bit [25] EXLEVEL_RL_EL1

Filter instruction trace for EL1 in Realm.

Case	Meaning
EXLEVEL_RL_EL1 == EXLEVEL_NS_EL1	The trace unit generates instruction trace for EL1 in Realm.
EXLEVEL_RL_EL1 != EXLEVEL_NS_EL1	The trace unit does not generate instruction trace for EL1 in Realm state.

On a Trace unit reset, this field resets to an architecturally unknown value.

Bit [26] EXLEVEL_RL_EL2

Filter instruction trace for EL2 in Realm.

Case	Meaning
EXLEVEL_RL_EL2 == EXLEVEL_NS_EL2	The trace unit generates instruction trace for EL2 in Realm.
EXLEVEL_RL_EL2 != EXLEVEL_NS_EL2	The trace unit does not generate instruction trace for EL2 in Realm state.

On a Trace unit reset, this field resets to an architecturally unknown value.

Chapter 16
AArch32 PE architectural state

16.1 System registers

16.1.1 PMCCFILTR

$\mathrm{R}_{\mathrm{DYBSC}}$ The following additional field is defined:

Bit [21] RLU, Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMCCFILTR.U bit, cycles in Realm EL0 are counted. Otherwise, events in Realm EL0 are not counted.

On a Warm reset, this field resets to an architecturally UNKNOWN value.

If FEAT_RME is not implemented, this field is RES0.

$\mathrm{I}_{\mathrm{HVNSG}}$ PMCCFILTR.RLU has the same definition as PMCCFILTR_EL0.RLU.

$\mathrm{I}_{\mathrm{XPYFH}}$ For PMCCFILTR_EL0, FEAT_RME also defines RLK and RLH fields. These are not defined in the AArch32 PMEVETYPER register, as Armv9-A does not support AArch32 at EL1 or EL2. The corresponding bit positions in PMEVETYPER are RES0.

16.1.2 PMEVTYPER<n>

$\mathrm{R}_{\mathrm{CHBSM}}$ The following additional field is defined:

Bit [21] RLU, Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>.U bit, events in Realm EL0 are counted. Otherwise, events in Realm EL0 are not counted.

On a Warm reset, this field resets to an architecturally UNKNOWN value.

If FEAT_RME is not implemented, this field is RES0.

I_{CXYTS} PMEVETYPER<n>.RLU has the same definition as PMEVETYPER<n>_EL0.RLU.

I_{JPXJG} For PMEVTYPER<n>_EL0, FEAT_RME also defines RLK and RLH fields. These are not defined in the AArch32 PMEVETYPER<n> register, as Armv9-A does not support AArch32 at EL1 or EL2. The corresponding bit positions in PMEVETYPER<n> are RES0.

Part A
Appendices

Chapter A1
Software usage examples

This chapter provides example software sequences for using the new features introduced by this specification.

A1.1 Granule Transition Flow

The properties of the Granule Transition Flow (GTF) are:

1. The GTF changes the PAS association of a physical granule from a previous physical address space to a new physical address space.
2. The GTF completes once the association of the physical granule with the new physical address space is observable.
3. When the GTF completes the following outcomes are guaranteed:
   a. Writes to the previous physical address space will not become observable.
   b. If the previous physical address space is Realm or Secure then no accesses, including Speculative read accesses, to the previous physical address space can observe unscrubbed values from before the GTF.
   c. If the previous physical address space is Realm or Secure then no instructions, including execution under Speculation, can observe unscrubbed values from before the GTF.
   d. If the previous physical address space is Realm or Secure then no accesses to the granule in the new physical address space observe unscrubbed values.
4. GTF outcomes for the new physical address space are guaranteed by EL3 without relying on cooperative behavior of SW that has access to the previous physical address space (for example, software running at EL2).

A1.1.1 Delegate

This sequence transitions the physical granule at address `addr` from Non-secure to Secure or Realm physical address space.

On implementations with FEAT_MTE2, Root firmware must issue `DC_CIGDPAPA` instead of `DC_CIPAPA`, in order to additionally clean and invalidate Allocation Tags associated with the affected locations.

```
Delegate(phys_addr* addr, PAS target_pas){

    // In order to maintain mutual distrust between Realm and Secure
    // states, remove any data speculatively fetched into the target
    // physical address space.
    for (i = 0; i<granule_size; i+=cache_line_size)
        DC_CIPAPA((addr+i), target_pas);

    DSB(OSH);

    write_gpt(addr, target_pas)
    DSB(OSHST);

    TLBI_RPALOS(addr, granule_size);
    DSB(OSH);

    for (i = 0; i<granule_size; i+=cache_line_size)
        DC_CIPAPA((addr+i), PAS_NS);

    DSB(OSH);
}
```

A1.1.2 Undelegate

This sequence transitions the physical granule at address `addr` from Secure or Realm physical address space to Non-secure.

The sequence assumes that the EL2 software for Secure or Realm state has already scrubbed the appropriate locations.

On implementations with FEAT_MTE2, Root firmware must issue `DC_CIGDPAPA` instead of `DC_CIPAPA`, in order to additionally clean and invalidate Allocation Tags associated with the affected locations.

```
Undelegate(phys_addr* addr, PAS current_pas)

    // In order to maintain mutual distrust between Realm and Secure
    // states, remove access now, in order to guarantee that writes
    // to the currently-accessible physical address space will not
    // later become observable.
    write_gpt(addr, No_access);
    DSB(OSHST);
    TLBI_RPALOS(addr, granule_size);
    DSB(OSH);

    // Ensure that the scrubbed data has made it past the PoPA
    for (i = 0; i<granule_size; i+=cache_line_size)
        DC_CIPAPA((addr+i), current_pas);

    DSB(OSH);

    // Remove any data loaded speculatively in NS space from before the scrubbing
    for (i = 0; i<granule_size; i+=cache_line_size)
        DC_CIPAPA((addr+i), PAS_NS);

    DSB(OSH);

    write_gpt(addr, PAS_NS);
    DSB(OSHST);

    // Ensure that all agents observe the new NS configuration
    TLBI_RPALOS(addr, granule_size);
    DSB(OSH);
```

A1.2 Procedures for changing the size of a GPT contiguous region

Example procedure to increase contiguity from 4KB to 2MB, assuming PGS is 4KB.

This example procedure does not consider mutual exclusion.

```
// Parameters:
// base = base address of desired contig region
// expected_gpi = value of all GPIs in the region

    assert IS_ALIGNED(base, 2MB);
    assert IS_VALID_GPI(expected_gpi);

    // Required GPTE is 16 GPI values, all the same
    uint64_t required_gpte;
    required_gpte = expected_gpi;
    required_gpte |= required_gpte << 4;
    required_gpte |= required_gpte << 8;
    required_gpte |= required_gpte << 16;
    required_gpte |= required_gpte << 32;

    for(gpte_addr=base, gpte_addr < base+2MB, gpte_addr += 64KB) {
        actual_gpte = gpt_entry(gpte_addr);
        // All entries must be consistent before the change
        if (actual_gpte != required_gpte)
            return false;
    }

    uint64_t new_gpte = 0x1;      // Contiguous descriptor
    new_gpte |= expected_gpi<<4; // GPI field
    new_gpte |= 0b01<<8;         // Contig field

    for(gpte_addr=base, gpte_addr < base+2MB, gpte_addr += 64KB) {
        set_gpt_entry(gpte_addr, new_gpte);
    }

    // No TLB maintenance required
    return true;
```

Example procedure to decrease contig from 2MB to 4KB, assuming PGS is 4KB.

This example procedure does not consider mutual exclusion.

```
// Parameters:
// base = base address of desired contig region
// expected_gpi = value of all GPIs in the region

    assert IS_ALIGNED(base, 2MB);
    assert IS_VALID_GPI(expected_gpi);

    // Required GPTE value
    uint64_t required_gpte = 0x1;      // Contiguous descriptor
    required_gpte |= expected_gpi<<4; // GPI field
    required_gpte |= 0b01<<8;         // Contig field

    for(gpte_addr=base, gpte_addr < base+2MB, gpte_addr += 64KB) {
        actual_gpte = gpt_entry(gpte_addr);
        // All entries must be consistent before the change
        if (actual_gpte != required_gpte)
            return false;
    }

    // New GPTE is 16 GPI values, all the same
    uint64_t new_gpte;
    new_gpte = expected_gpi;
    new_gpte |= new_gpte << 4;
    new_gpte |= new_gpte << 8;
    new_gpte |= new_gpte << 16;
    new_gpte |= new_gpte << 32;

    for(gpte_addr=base, gpte_addr < base+2MB, gpte_addr += 64KB) {
        set_gpt_entry(gpte_addr, new_gpte);
    }

    // It is assumed that the entries are being cracked so that they can
    // be changed, for whatever reason. In which case it required to
    // perform TLB maintenance now.
    DSB();
    TLBI_RPALOS(base, 2MB);
```

```
    DSB();
    return true;
```

Chapter A2
List of registers

This section provides the full information for registers added or modified by RME.

A2.1 AArch64 registers

A2.1.1 CNTHCTL_EL2, Counter-timer Hypervisor Control register

The CNTHCTL_EL2 characteristics are:

Purpose

Controls the generation of an event stream from the physical counter, and access from EL1 to the physical counter and the EL1 physical timer.

Attributes

CNTHCTL_EL2 is a 64-bit register.

Configuration

If EL2 is not implemented, this register is RES0 from EL3.

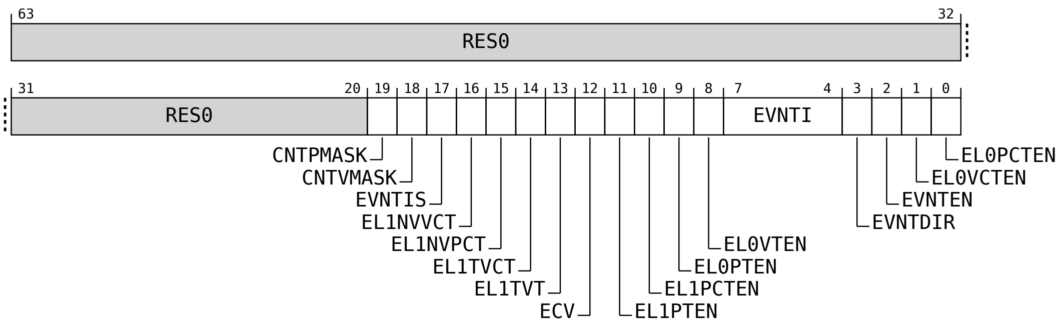
This register has no effect if EL2 is not enabled in the current Security state.

AArch64 system register CNTHCTL_EL2 bits [31:0] are architecturally mapped to AArch32 system register CNTHCTL[31:0].

Field descriptions

The CNTHCTL_EL2 bit assignments are:

When FEAT_VHE is implemented and HCR_EL2.E2H == 1:



Bits [63:20]

Reserved, RES0.

CNTPMASK, bit [19]

When FEAT_RME is implemented:

Value	Meaning
`0b0`	This control has no affect on CNTP_CTL_EL0.IMASK.
`0b1`	CNTP_CTL_EL0.IMASK behaves as if set to 1 for all purposes other than a direct read of the field.

This bit is RES0 in Non-secure and Secure state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

CNTVMASK, bit [18]

When FEAT_RME is implemented:

Value	Meaning
0b0	This control has no affect on CNTV_CTL_EL0.IMASK.
0b1	CNTV_CTL_EL0.IMASK behaves as if set to 1 for all purposes other than a direct read of the field.

This bit is RES0 in Non-secure and Secure state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EVNTIS, bit [17]

When FEAT_ECV is implemented:

Controls the scale of the generation of the event stream.

Value	Meaning
0b0	The CNTHCTL_EL2.EVNTI field applies to CNTPCT_EL0[15:0].
0b1	The CNTHCTL_EL2.EVNTI field applies to CNTPCT_EL0[23:8].

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EL1NVVCT, bit [16]

When FEAT_ECV is implemented:

Traps EL1 accesses to the specified EL1 virtual timer registers using the EL02 descriptors to EL2, when EL2 is enabled for the current Security state.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.

Value	Meaning
0b1	If ((HCR_EL2.E2H==1 && HCR_EL2.TGE==1) || HCR_EL2.NV2==0 || HCR_EL2.NV1==1 || HCR_EL2.NV==0), this control does not cause any instructions to be trapped. If ((HCR_EL2.E2H==0 || HCR_EL2.TGE==0) && HCR_EL2.NV2==1 && HCR_EL2.NV1==0 && HCR_EL2.NV==1), then EL1 accesses to CNTV_CTL_EL02 and CNTV_CVAL_EL02 are trapped to EL2.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EL1NVPCT, bit [15]

When FEAT_ECV is implemented:

Traps EL1 accesses to the specified EL1 physical timer registers using the EL02 descriptors to EL2, when EL2 is enabled for the current Security state.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	If ((HCR_EL2.E2H==1 && HCR_EL2.TGE==1) || HCR_EL2.NV2==0 || HCR_EL2.NV1==1 || HCR_EL2.NV==0), this control does not cause any instructions to be trapped. If (HCR_EL2.E2H==0 || HCR_EL2.TGE==0) && HCR_EL2.NV2==1 && HCR_EL2.NV1==0 && HCR_EL2.NV==1, then EL1 accesses to CNTP_CTL_EL02 and CNTP_CVAL_EL02, are trapped to EL2.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EL1TVCT, bit [14]

When FEAT_ECV is implemented:

Traps EL0 and EL1 accesses to the EL1 virtual counter registers to EL2, when EL2 is enabled for the current Security state.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	If HCR_EL2.{E2H, TGE} is {1, 1}, this control does not cause any instructions to be trapped. If HCR_EL2.E2H is 0 or HCR_EL2.TGE is 0, then: • In AArch64 state, traps EL0 and EL1 accesses to CNTVCT_EL0 to EL2, unless they are trapped by CNTKCTL_EL1.EL0VCTEN. • In AArch32 state, traps EL0 and EL1 accesses to CNTVCT to EL2, unless they are trapped by CNTKCTL_EL1.EL0VCTEN or CNTKCTL.PL0VCTEN.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EL1TVT, bit [13]

When FEAT_ECV is implemented:

Traps EL0 and EL1 accesses to the EL1 virtual timer registers to EL2, when EL2 is enabled for the current Security state.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.

Value	Meaning
`0b1`	If HCR_EL2.{E2H, TGE} is {1, 1}, this control does not cause any instructions to be trapped. If HCR_EL2.E2H is 0 or HCR_EL2.TGE is 0, then: • In AArch64 state, traps EL0 and EL1 accesses to CNTV_CTL_EL0, CNTV_CVAL_EL0, and CNTV_TVAL_EL0 to EL2, unless they are trapped by CNTKCTL_EL1.EL0VTEN. • In AArch32 state, traps EL0 and EL1 accesses to CNTV_CTL, CNTV_CVAL, and CNTV_TVAL to EL2, unless they are trapped by CNTKCTL_EL1.EL0VTEN or CNTKCTL.PL0VTEN.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

ECV, bit [12]

When FEAT_ECV is implemented:

Enables the Enhanced Counter Virtualization functionality registers.

Value	Meaning
`0b0`	Enhanced Counter Virtualization functionality is disabled.
`0b1`	When HCR_EL2.{E2H, TGE} == {1, 1} or SCR_EL3.{NS, EEL2} == {0, 0}, then Enhanced Counter Virtualization functionality is disabled. When SCR_EL3.NS or SCR_EL3.EEL2 are 1, and HCR_EL2.E2H or HCR_EL2.TGE are 0, then Enhanced Counter Virtualziation functionality is enabled when EL2 is enabled for the current Security state. This means that: • An MRS to CNTPCT_EL0 from either EL0 or EL1 that is not trapped will return the value (PCount<63:0> - CNTPOFF_EL2<63:0>). • The EL1 physical timer interrupt is triggered when ((PCount<63:0> - CNTPOFF_EL2<63:0>) - PCVal<63:0>) is greater than or equal to 0. PCount<63:0> is the physical count returned when CNTPCT_EL0 is read from EL2 or EL3. PCVal<63:0> is the EL1 physical timer compare value for this timer.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EL1PTEN, bit [11]

When HCR_EL2.TGE is 0, traps EL0 and EL1 accesses to the E1 physical timer registers to EL2 when EL2 is enabled in the current Security state.

Value	Meaning
0b0	From AArch64 state: EL0 and EL1 accesses to the CNTP_CTL_EL0, CNTP_CVAL_EL0, and CNTP_TVAL_EL0 are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PTEN. From AArch32 state: EL0 and EL1 accesses to the CNTP_CTL, CNTP_CVAL, and CNTP_TVAL are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PTEN or CNTKCTL.PL0PTEN.
0b1	This control does not cause any instructions to be trapped.

When HCR_EL2.TGE is 1, this control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EL1PCTEN, bit [10]

When HCR_EL2.TGE is 0, traps EL0 and EL1 accesses to the EL1 physical counter register to EL2 when EL2 is enabled in the current Security state, as follows:

- In AArch64 state, accesses to CNTPCT_EL0 are trapped to EL2, reported using EC syndrome value 0x18.
- In AArch32 state, MRRC or MCRR accesses to CNTPCT are trapped to EL2, reported using EC syndrome value 0x04.

Value	Meaning
0b0	From AArch64 state: EL0 and EL1 accesses to the CNTPCT_EL0 are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PCTEN. From AArch32 state: EL0 and EL1 accesses to the CNTPCT are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PCTEN or CNTKCTL.PL0PCTEN.
0b1	This control does not cause any instructions to be trapped.

When HCR_EL2.TGE is 1, this control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EL0PTEN, bit [9]

When HCR_EL2.TGE is 0, this control does not cause any instructions to be trapped.

When HCR_EL2.TGE is 1, traps EL0 accesses to the physical timer registers to EL2.

Value	Meaning
`0b0`	EL0 using AArch64: EL0 accesses to the CNTP_CTL_EL0, CNTP_CVAL_EL0, and CNTP_TVAL_EL0 registers are trapped to EL2. EL0 using AArch32: EL0 accesses to the CNTP_CTL, CNTP_CVAL and CNTP_TVAL registers are trapped to EL2.
`0b1`	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EL0VTEN, bit [8]

When HCR_EL2.TGE is 0, this control does not cause any instructions to be trapped.

When HCR_EL2.TGE is 1, traps EL0 accesses to the virtual timer registers to EL2.

Value	Meaning
`0b0`	EL0 using AArch64: EL0 accesses to the CNTV_CTL_EL0, CNTV_CVAL_EL0, and CNTV_TVAL_EL0 registers are trapped to EL2. EL0 using AArch32: EL0 accesses to the CNTV_CTL, CNTV_CVAL, and CNTV_TVAL registers are trapped to EL2.
`0b1`	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EVNTI, bits [7:4]

Selects which bit of CNTPCT_EL0, as seen from EL2, is the trigger for the event stream generated from that counter when that stream is enabled.

If FEAT_ECV is implemented, and CNTHCTL_EL2.EVNTIS is 1, this field selects a trigger bit in the range 8 to 23 of CNTPCT_EL0.

Otherwise, this field selects a trigger bit in the range 0 to 15 of CNTPCT_EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EVNTDIR, bit [3]

Controls which transition of the CNTPCT_EL0 trigger bit, as seen from EL2 and defined by EVNTI, generates an event when the event stream is enabled.

Value	Meaning
0b0	A 0 to 1 transition of the trigger bit triggers an event.
0b1	A 1 to 0 transition of the trigger bit triggers an event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EVNTEN, bit [2]

Enables the generation of an event stream from CNTPCT_EL0 as seen from EL2.

Value	Meaning
0b0	Disables the event stream.
0b1	Enables the event stream.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EL0VCTEN, bit [1]

When HCR_EL2.TGE is 0, this control does not cause any instructions to be trapped.

When HCR_EL2.TGE is 1, traps EL0 accesses to the frequency register and virtual counter register to EL2.

Value	Meaning
0b0	EL0 using AArch64: EL0 accesses to the CNTVCT_EL0 are trapped to EL2. EL0 using AArch64: EL0 accesses to the CNTFRQ_EL0 register are trapped to EL2, if CNTHCTL_EL2.EL0PCTEN is also 0. EL0 using AArch32: EL0 accesses to the CNTVCT are trapped to EL2. EL0 using AArch32: EL0 accesses to the CNTFRQ register are trapped to EL2, if CNTHCTL.EL0PCTEN is also 0.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EL0PCTEN, bit [0]

When HCR_EL2.TGE is 0, this control does not cause any instructions to be trapped.

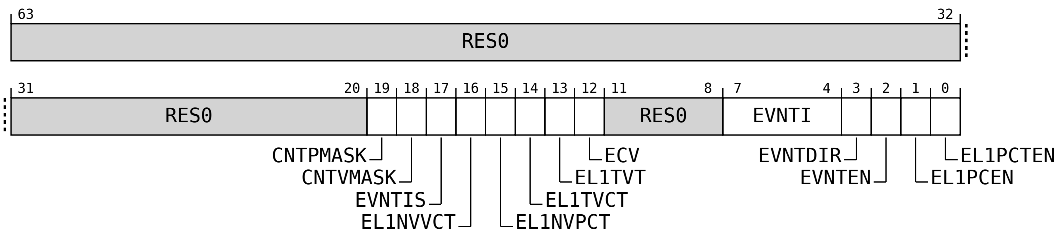
When HCR_EL2.TGE is 1, traps EL0 accesses to the frequency register and physical counter register to EL2.

Value	Meaning
`0b0`	EL0 using AArch64: EL0 accesses to the CNTPCT_EL0 are trapped to EL2. EL0 using AArch64: EL0 accesses to the CNTFRQ_EL0 register are trapped to EL2, if CNTHCTL_EL2.EL0VCTEN is also 0. EL0 using AArch32: EL0 accesses to the CNTPCT are trapped to EL2. EL0 using AArch32: EL0 accesses to the CNTFRQ and register are trapped to EL2, if CNTHCTL_EL2.EL0VCTEN is also 0.
`0b1`	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:



This format applies in all Armv8.0 implementations, and it also contains a description of the behavior when EL3 is implemented and EL2 is not implemented.

Bits [63:20]

Reserved, RES0.

CNTPMASK, bit [19]

When FEAT_RME is implemented:

Value	Meaning
`0b0`	This control has no affect on CNTP_CTL_EL0.IMASK.
`0b1`	CNTP_CTL_EL0.IMASK behaves as if set to 1 for all purposes other than a direct read of the field.

This bit is RES0 in Non-secure and Secure state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

CNTVMASK, bit [18]

When FEAT_RME is implemented:

Value	Meaning
0b0	This control has no affect on CNTV_CTL_EL0.IMASK.
0b1	CNTV_CTL_EL0.IMASK behaves as if set to 1 for all purposes other than a direct read of the field.

This bit is RES0 in Non-secure and Secure state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EVNTIS, bit [17]

When FEAT_ECV is implemented:

Controls the scale of the generation of the event stream.

Value	Meaning
0b0	The CNTHCTL_EL2.EVNTI field applies to CNTPCT_EL0[15:0].
0b1	The CNTHCTL_EL2.EVNTI field applies to CNTPCT_EL0[23:8].

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EL1NVVCT, bit [16]

When FEAT_ECV is implemented:

Traps EL1 accesses to the specified EL1 virtual timer registers using the EL02 descriptors to EL2, when EL2 is enabled for the current Security state.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	If ((HCR_EL2.E2H==1 && HCR_EL2.TGE==1) || HCR_EL2.NV2==0 || HCR_EL2.NV1==1 || HCR_EL2.NV==0), this control does not cause any instructions to be trapped. If ((HCR_EL2.E2H==0 || HCR_EL2.TGE==0) && HCR_EL2.NV2==1 && HCR_EL2.NV1==0 && HCR_EL2.NV==1), then EL1 accesses to CNTV_CTL_EL02 and CNTV_CVAL_EL02 are trapped to EL2.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EL1NVPCT, bit [15]

When FEAT_ECV is implemented:

Traps EL1 accesses to the specified EL1 physical timer registers using the EL02 descriptors to EL2, when EL2 is enabled for the current Security state.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	If ((HCR_EL2.E2H==1 && HCR_EL2.TGE==1) || HCR_EL2.NV2==0 || HCR_EL2.NV1==1 || HCR_EL2.NV==0), this control does not cause any instructions to be trapped. If (HCR_EL2.E2H==0 || HCR_EL2.TGE==0) && HCR_EL2.NV2==1 && HCR_EL2.NV1==0 && HCR_EL2.NV==1, then EL1 accesses to CNTP_CTL_EL02 and CNTP_CVAL_EL02, are trapped to EL2.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EL1TVCT, bit [14]

When FEAT_ECV is implemented:

Traps EL0 and EL1 accesses to the EL1 virtual counter registers to EL2, when EL2 is enabled for the current Security state.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	If HCR_EL2.{E2H, TGE} is {1, 1}, this control does not cause any instructions to be trapped. If HCR_EL2.E2H is 0 or HCR_EL2.TGE is 0, then: In AArch64 state, traps EL0 and EL1 accesses to CNTVCT_EL0 to EL2, unless they are trapped by CNTKCTL_EL1.EL0VCTEN. In AArch32 state, traps EL0 and EL1 accesses to CNTVCT to EL2, unless they are trapped by CNTKCTL_EL1.EL0VCTEN or CNTKCTL.PL0VCTEN.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EL1TVT, bit [13]

When FEAT_ECV is implemented:

Traps EL0 and EL1 accesses to the EL1 virtual timer registers to EL2, when EL2 is enabled for the current Security state.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.

Value	Meaning
`0b1`	If HCR_EL2.{E2H, TGE} is {1, 1}, this control does not cause any instructions to be trapped. If HCR_EL2.E2H is 0 or HCR_EL2.TGE is 0, then: • In AArch64 state, traps EL0 and EL1 accesses to CNTV_CTL_EL0, CNTV_CVAL_EL0, and CNTV_TVAL_EL0 to EL2, unless they are trapped by CNTKCTL_EL1.EL0VTEN. • In AArch32 state, traps EL0 and EL1 accesses to CNTV_CTL, CNTV_CVAL, and CNTV_TVAL to EL2, unless they are trapped by CNTKCTL_EL1.EL0VTEN or CNTKCTL.PL0VTEN.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

ECV, bit [12]

When FEAT_ECV is implemented:

Enables the Enhanced Counter Virtualization functionality registers.

Value	Meaning
`0b0`	Enhanced Counter Virtualization functionality is disabled.
`0b1`	When HCR_EL2.{E2H, TGE} == {1, 1} or SCR_EL3.{NS, EEL2} == {0, 0}, then Enhanced Counter Virtualization functionality is disabled. When SCR_EL3.NS or SCR_EL3.EEL2 are 1, and HCR_EL2.E2H or HCR_EL2.TGE are 0, then Enhanced Counter Virtualziation functionality is enabled when EL2 is enabled for the current Security state. This means that: • An MRS to CNTPCT_EL0 from either EL0 or EL1 that is not trapped will return the value (PCount<63:0> - CNTPOFF_EL2<63:0>). • The EL1 physical timer interrupt is triggered when ((PCount<63:0> - CNTPOFF_EL2<63:0>) - PCVal<63:0>) is greater than or equal to 0. PCount is the physical count returned when CNTPCT_EL0 is read from EL2 or EL3. PCVal<63:0> is the EL1 physical timer compare value for this timer.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [11:8]

Reserved, RES0.

EVNTI, bits [7:4]

Selects which bit of CNTPCT_EL0, as seen from EL2,is the trigger for the event stream generated from that counter when that stream is enabled.

If FEAT_ECV is implemented, and CNTHCTL_EL2.EVNTIS is 1, this field selects a trigger bit in the range 8 to 23 of CNTPCT_EL0.

Otherwise, this field selects a trigger bit in the range 0 to 15 of CNTPCT_EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EVNTDIR, bit [3]

Controls which transition of the CNTPCT_EL0 trigger bit, as seen from EL2 and defined by EVNTI, generates an event when the event stream is enabled.

Value	Meaning
0b0	A 0 to 1 transition of the trigger bit triggers an event.
0b1	A 1 to 0 transition of the trigger bit triggers an event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EVNTEN, bit [2]

Enables the generation of an event stream from CNTPCT_EL0 as seen from EL2.

Value	Meaning
0b0	Disables the event stream.
0b1	Enables the event stream.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EL1PCEN, bit [1]

Traps EL0 and EL1 accesses to the EL1 physical timer registers to EL2 when EL2 is enabled in the current Security state, as follows:

- In AArch64 state, accesses to CNTP_CTL_EL0, CNTP_CVAL_EL0, CNTP_TVAL_EL0 are trapped to EL2, reported using EC syndrome value 0x18.
- In AArch32 state, MRC or MCR accesses to the following registers are trapped to EL2 reported using EC syndrome value 0x3 and MRRC and MCRR accesses are trapped to EL2, reported using EC syndrome value 0x04:
  – CNTP_CTL, CNTP_CVAL, CNTP_TVAL.

Value	Meaning
`0b0`	From AArch64 state: EL0 and EL1 accesses to the CNTP_CTL_EL0, CNTP_CVAL_EL0, and CNTP_TVAL_EL0 are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PTEN. From AArch32 state: EL0 and EL1 accesses to the CNTP_CTL, CNTP_CVAL, and CNTP_TVAL are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PTEN or CNTKCTL.PL0PTEN.
`0b1`	This control does not cause any instructions to be trapped.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 1 other than for the purpose of a direct read.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EL1PCTEN, bit [0]

Traps EL0 and EL1 accesses to the EL1 physical counter register to EL2 when EL2 is enabled in the current Security state, as follows:

- In AArch64 state, accesses to CNTPCT_EL0 are trapped to EL2, reported using EC syndrome value 0x18.
- In AArch32 state, MRRC or MCRR accesses to CNTPCT are trapped to EL2, reported using EC syndrome value 0x04.

Value	Meaning
`0b0`	From AArch64 state: EL0 and EL1 accesses to the CNTPCT_EL0 are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PCTEN. From AArch32 state: EL0 and EL1 accesses to the CNTPCT are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PCTEN or CNTKCTL.PL0PCTEN.
`0b1`	This control does not cause any instructions to be trapped.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 1 other than for the purpose of a direct read.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing the CNTHCTL_EL2

When HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic CNTHCTL_EL2 or CNTKCTL_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, CNTHCTL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return CNTHCTL_EL2;
elsif PSTATE.EL == EL3 then
    return CNTHCTL_EL2;
```

MSR CNTHCTL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    CNTHCTL_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    CNTHCTL_EL2 = X[t];
```

MRS <Xt>, CNTKCTL_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1110	0b0001	0b000

```
if PSTATE.EL == EL0 then
```

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    return CNTKCTL_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return CNTHCTL_EL2;
    else
        return CNTKCTL_EL1;
elsif PSTATE.EL == EL3 then
    return CNTKCTL_EL1;
```

MSR CNTKCTL_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1110	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    CNTKCTL_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        CNTHCTL_EL2 = X[t];
    else
        CNTKCTL_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    CNTKCTL_EL1 = X[t];
```

A2.1.2 DBGAUTHSTATUS_EL1, Debug Authentication Status register

The DBGAUTHSTATUS_EL1 characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

Attributes

DBGAUTHSTATUS_EL1 is a 64-bit register.

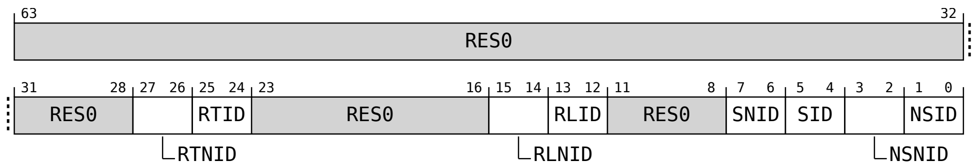
Configuration

AArch64 system register DBGAUTHSTATUS_EL1 bits [31:0] are architecturally mapped to AArch32 system register DBGAUTHSTATUS[31:0].

AArch64 system register DBGAUTHSTATUS_EL1 bits [31:0] are architecturally mapped to External register DBGAUTHSTATUS_EL1[31:0].

Field descriptions

The DBGAUTHSTATUS_EL1 bit assignments are:



Bits [63:28]

Reserved, RES0.

RTNID, bits [27:26]

Root non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RTID.

RTID, bits [25:24]

Root invasive debug.

Value	Meaning
`0b00`	Not implemented.
`0b10`	Implemented and disabled. ExternalRootInvasiveDebugEnabled() == FALSE.
`0b11`	Implemented and enabled. ExternalRootInvasiveDebugEnabled() == TRUE.

All other values are reserved.

If FEAT_RME is not implemented, the only permitted value is 00.

Bits [23:16]

Reserved, RES0.

RLNID, bits [15:14]

Realm non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RLID.

RLID, bits [13:12]

Realm invasive debug.

Value	Meaning
`0b00`	Not implemented.
`0b10`	Implemented and disabled. ExternalRealmInvasiveDebugEnabled() == FALSE.
`0b11`	Implemented and enabled. ExternalRealmInvasiveDebugEnabled() == TRUE.

All other values are reserved.

If FEAT_RME is not implemented, the only permitted value is 00.

Bits [11:8]

Reserved, RES0.

Bits [7:6]

When FEAT_Debugv8p4 is implemented

SNID, bits [7:6]

Secure non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.SID.

Otherwise

SNID, bits [7:6]

Secure non-invasive debug.

Value	Meaning
`0b00`	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 1.
`0b10`	Implemented and disabled. ExternalSecureNoninvasiveDebugEnabled() == FALSE.
`0b11`	Implemented and enabled. ExternalSecureNoninvasiveDebugEnabled() == TRUE.

All other values are reserved.

SID, bits [5:4]

Secure invasive debug.

Value	Meaning
`0b00`	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 1.
`0b10`	Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE.
`0b11`	Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.

All other values are reserved.

Bits [3:2]

When FEAT_Debugv8p4 is implemented

NSNID, bits [3:2]

Non-secure non-invasive debug.

Value	Meaning
`0b00`	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
`0b11`	Implemented and enabled. EL3 is implemented or the Effective value of SCR_EL3.NS is 1.

All other values are reserved.

Otherwise

NSNID, bits [3:2]

Non-secure non-invasive debug.

Value	Meaning
`0b00`	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
`0b10`	Implemented and disabled. ExternalNoninvasiveDebugEnabled() == FALSE.
`0b11`	Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE.

All other values are reserved.

NSID, bits [1:0]

Non-secure invasive debug.

Value	Meaning
0b00	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
0b10	Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE.

All other values are reserved.

Accessing the DBGAUTHSTATUS_EL1

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, DBGAUTHSTATUS_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1110	0b110

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
      ↪when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.DBGAUTHSTATUS_EL1 == '1'
      ↪then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return DBGAUTHSTATUS_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
      ↪when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return DBGAUTHSTATUS_EL1;
elsif PSTATE.EL == EL3 then
    return DBGAUTHSTATUS_EL1;
```

A2.1.3 DBGBCR<n>_EL1, Debug Breakpoint Control Registers, n = 0 - 15

The DBGBCR<n>_EL1 characteristics are:

Purpose

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR<n>_EL1.

Attributes

DBGBCR<n>_EL1 is a 64-bit register.

Configuration

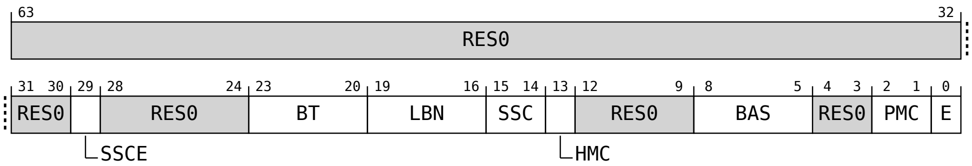
If breakpoint n is not implemented, accesses to this register are UNDEFINED.

AArch64 system register DBGBCR<n>_EL1 bits [31:0] are architecturally mapped to AArch32 system register DBGBCR<n>[31:0].

AArch64 system register DBGBCR<n>_EL1 bits [31:0] are architecturally mapped to External register DBGBCR<n>_EL1[31:0].

Field descriptions

The DBGBCR<n>_EL1 bit assignments are:



Bits [63:30]

Reserved, RES0.

SSCE, bit [29]

When FEAT_RME is implemented:

Security State Control Extended.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [28:24]

Reserved, RES0.

BT, bits [23:20]

Breakpoint Type. Possible values are:

Value	Meaning
`0b0000`	Unlinked instruction address match. DBGBVR<n>_EL1 is the address of an instruction.
`0b0001`	As 0b0000, but linked to a Context matching breakpoint.
`0b0010`	Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of HCR_EL2.E2H is 1, if either the PE is executing at EL0 with HCR_EL2.TGE set to 1 or the PE is executing at EL2, then DBGBVR<n>_EL1.ContextID must match the CONTEXTIDR_EL2 value. Otherwise, DBGBVR<n>_EL1.ContextID must match the CONTEXTIDR_EL1 value
`0b0011`	As 0b0010, with linking enabled.
`0b0110`	Unlinked CONTEXTIDR_EL1 match. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.
`0b0111`	As 0b0110, with linking enabled.
`0b1000`	Unlinked VMID match. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.
`0b1001`	As 0b1000, with linking enabled.
`0b1010`	Unlinked VMID and Context ID match. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.
`0b1011`	As 0b1010, with linking enabled.
`0b1100`	Unlinked CONTEXTIDR_EL2 match. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.
`0b1101`	As 0b1100, with linking enabled.
`0b1110`	Unlinked Full Context ID match. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2.
`0b1111`	As 0b1110, with linking enabled.

All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see ‘Execution conditions for which a breakpoint generates Breakpoint exceptions’.

For more information on the effect of programming the fields to a reserved value, see ’Reserved DBGBCR<n>_EL1.BT values’.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LBN, bits [19:16]

Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to.

For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value.

This field is ignored when the value of DBGBCR<n>_EL1.E is 0.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

For more information on the effect of programming the fields to a reserved set of values, see 'Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

For more information, see DBGBCR<n>_EL1.SSC.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [12:9]

Reserved, RES0.

BAS, bits [8:5]

When AArch32 is supported:

Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state.

The permitted values depend on the breakpoint type.

For Address match breakpoints, the permitted values are:

BAS	Match instruction at	Constraint for debuggers
0b0011	DBGBVR<n>_EL1	Use for T32 instructions
0b1100	DBGBVR<n>_EL1 + 2	Use for T32 instructions
0b1111	DBGBVR<n>_EL1	Use for A64 and A32 instructions

All other values are reserved. For more information, see 'Reserved DBGBCR<n>_EL1.BAS values'.

For more information on using the BAS field in address match breakpoints, see 'Using the BAS field in Address Match breakpoints'.

For Context matching breakpoints, this field is RES1 and ignored.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES1

Bits [4:3]

Reserved, RES0.

PMC, bits [2:1]

Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

For more information, see DBGBCR<n>_EL1.SSC.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

E, bit [0]

Enable breakpoint DBGBVR<n>_EL1.

Value	**Meaning**
`0b0`	Breakpoint disabled.
`0b1`	Breakpoint enabled.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing the DBGBCR<n>_EL1

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, DBGBCR<n>_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	n[3:0]	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        return DBGBCR_EL1[UInt(CRm<3:0>)];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        return DBGBCR_EL1[UInt(CRm<3:0>)];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        return DBGBCR_EL1[UInt(CRm<3:0>)];
```

MSR DBGBCR<n>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	n[3:0]	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[UInt(CRm<3:0>)] = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[UInt(CRm<3:0>)] = X[t];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[UInt(CRm<3:0>)] = X[t];
```

A2.1.4 DBGWCR<n>_EL1, Debug Watchpoint Control Registers, n = 0 - 15

The DBGWCR<n>_EL1 characteristics are:

Purpose

Holds control information for a watchpoint. Forms watchpoint n together with value register DBGWVR<n>_EL1.

Attributes

DBGWCR<n>_EL1 is a 64-bit register.

Configuration

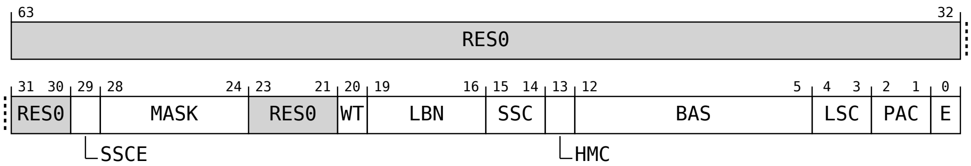
If watchpoint n is not implemented then accesses to this register are UNDEFINED.

AArch64 system register DBGWCR<n>_EL1 bits [31:0] are architecturally mapped to AArch32 system register DBGWCR<n>[31:0].

AArch64 system register DBGWCR<n>_EL1 bits [31:0] are architecturally mapped to External register DBGWCR<n>_EL1[31:0].

Field descriptions

The DBGWCR<n>_EL1 bit assignments are:



Bits [63:30]

Reserved, RES0.

SSCE, bit [29]

When FEAT_RME is implemented:

Security State Control Extended.

The fields that indicate when the watchpoint can be generated are: HMC, PAC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

MASK, bits [28:24]

Address mask. Only objects up to 2GB can be watched using a single mask.

Value	Meaning
0b00000	No mask.

Value	Meaning
0b00001	Reserved.
0b00010	Reserved.

If programmed with a reserved value, a watchpoint must behave as if either:

- MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1.
- The watchpoint is disabled.

Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture.

Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [23:21]

Reserved, RES0.

WT, bit [20]

Watchpoint type. Possible values are:

Value	Meaning
0b0	Unlinked data address match.
0b1	Linked data address match.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LBN, bits [19:16]

Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated.

The fields that indicate when the watchpoint can be generated are: HMC, PAC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see ‘Execution conditions for which a watchpoint generates Watchpoint exceptions’.

For more information on the effect of programming the fields to a reserved value, see 'Reserved DBGWCR<n>_EL1.{SSC, HMC, PAC} values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated.

The fields that indicate when the watchpoint can be generated are: HMC, PAC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

BAS, bits [12:5]

Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR<n>_EL1 is being watched.

BAS	Description
xxxxxxx1	Match byte at DBGWVR<n>_EL1
xxxxxx1x	Match byte at DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at DBGWVR<n>_EL1 + 2
xxxx1xxx	Match byte at DBGWVR<n>_EL1 + 3

In cases where DBGWVR<n>_EL1 addresses a double-word:

BAS	Description, if DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at DBGWVR<n>_EL1 + 7

If DBGWVR<n>_EL1[2] == 1, only BAS[3:0] are used and BAS[7:4] are ignored. Arm deprecates setting DBGWVR<n>_EL1[2] == 1.

The valid values for BAS are non-zero binary numbers all of whose set bits are contiguous. All other values are reserved and must not be used by software. See 'Reserved DBGWCR<n>_EL1.BAS values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LSC, bits [4:3]

Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are:

Value	Meaning
`0b01`	Match instructions that load from a watchpointed address.
`0b10`	Match instructions that store to a watchpointed address.
`0b11`	Match instructions that load from or store to a watchpointed address.

All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

PAC, bits [2:1]

Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated.

The fields that indicate when the watchpoint can be generated are: HMC, PAC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

E, bit [0]

Enable watchpoint n. Possible values are:

Value	Meaning
`0b0`	Watchpoint disabled.
`0b1`	Watchpoint enabled.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing the DBGWCR<n>_EL1

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, DBGWCR<n>_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	n[3:0]	0b111

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
        ↪when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        return DBGWCR_EL1[UInt(CRm<3:0>)];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
        ↪when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        return DBGWCR_EL1[UInt(CRm<3:0>)];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        return DBGWCR_EL1[UInt(CRm<3:0>)];
```

MSR DBGWCR<n>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	n[3:0]	0b111

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
        ↪when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[UInt(CRm<3:0>)] = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
```

```
           ↪when SDD == '1'" && MDCR_EL3.TDA == '1' then
            UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[UInt(CRm<3:0>)] = X[t];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[UInt(CRm<3:0>)] = X[t];
```

A2.1.5 ESR_EL1, Exception Syndrome Register (EL1)

The ESR_EL1 characteristics are:

Purpose

Holds syndrome information for an exception taken to EL1.

Attributes

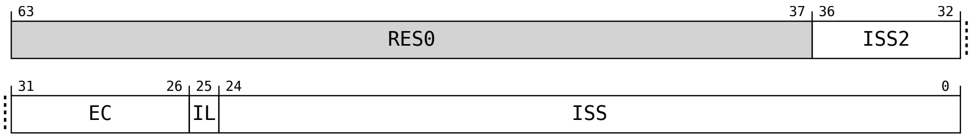
ESR_EL1 is a 64-bit register.

Configuration

AArch64 system register ESR_EL1 bits [31:0] are architecturally mapped to AArch32 system register DFSR[31:0].

Field descriptions

The ESR_EL1 bit assignments are:



ESR_EL1 is made UNKNOWN as a result of an exception return from EL1.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to EL1, the value of ESR_EL1 is UNKNOWN. The value written to ESR_EL1 must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not UNPREDICTABLE at that Exception level, in order to avoid the possibility of a privilege violation.

Bits [63:37]

Reserved, RES0.

ISS2, bits [36:32]

When FEAT_LS64 is implemented:

If a memory access generated by an ST64BV or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

For any other Data Abort, this field is RES0.

Otherwise:

RES0

EC, bits [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about.

For each EC value, the table references a subsection that gives information about:

- The cause of the exception, for example the configuration required to enable the trap.
- The encoding of the associated ISS.

Possible values of the EC field are:

Value	Meaning	Link	Applies
`0b000000`	Unknown reason.	ISS - exceptions with an unknown reason	
`0b000001`	Trapped WF* instruction execution. Conditional WF* instructions that fail their condition code check do not cause an exception.	ISS - an exception from a WF* instruction	
`0b000011`	Trapped MCR or MRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS - an exception from an MCR or MRC access	When AArch32 is supported
`0b000100`	Trapped MCRR or MRRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS - an exception from an MCRR or MRRC access	When AArch32 is supported
`0b000101`	Trapped MCR or MRC access with (coproc==0b1110).	ISS - an exception from an MCR or MRC access	When AArch32 is supported
`0b000110`	Trapped LDC or STC access. The only architected uses of these instruction are: • An STC to write data to memory from DBGDTRRXint. • An LDC to read data from memory to DBGDTRTXint.	ISS - an exception from an LDC or STC instruction	When AArch32 is supported
`0b000111`	Access to SME, SVE, Advanced SIMD or floating-point functionality trapped by CPACR_EL1.FPEN, CPTR_EL2.FPEN, CPTR_EL2.TFP, or CPTR_EL3.TFP control. Excludes exceptions resulting from CPACR_EL1 when the value of HCR_EL2.TGE is 1, or because SVE or Advanced SIMD and floating-point are not implemented. These are reported with EC value 0b000000 as described in 'The EC used to report an exception routed to EL2 because HCR_EL2.TGE is 1'.	ISS - an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps	
`0b001010`	Trapped execution of an LD64B, ST64B, ST64BV, or ST64BV0 instruction.	ISS - an exception from an LD64B or ST64B* instruction	When FEAT_LS64 is implemented
`0b001100`	Trapped MRRC access with (coproc==0b1110).	ISS - an exception from an MCRR or MRRC access	When AArch32 is supported
`0b001101`	Branch Target Exception.	ISS - an exception from Branch Target Identification instruction	When FEAT_BTI is implemented
`0b001110`	Illegal Execution state.	ISS - an exception from an Illegal Execution state, or a PC or SP alignment fault	
`0b010001`	SVC instruction execution in AArch32 state.	ISS - an exception from HVC or SVC instruction execution	When AArch32 is supported
`0b010101`	SVC instruction execution in AArch64 state.	ISS - an exception from HVC or SVC instruction execution	When AArch64 is supported

Value	Meaning	Link	Applies
0b011000	Trapped MSR, MRS or System instruction execution in AArch64 state, that is not reported using EC 0b000000, 0b000001, or 0b000111. This includes all instructions that cause exceptions that are part of the encoding space defined in 'System instruction class encoding overview', except for those exceptions reported using EC values 0b000000, 0b000001, or 0b000111.	ISS - an exception from MSR, MRS, or System instruction execution in AArch64 state	When AArch64 is supported
0b011001	Access to SVE functionality trapped as a result of CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ, that is not reported using EC 0b000000.	ISS - an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ	When FEAT_SVE is implemented
0b011011	Exception from an access to a TSTART instruction at EL0 when SCTLR_EL1.TME0 == 0, EL0 when SCTLR_EL2.TME0 == 0, at EL1 when SCTLR_EL1.TME == 0, at EL2 when SCTLR_EL2.TME == 0 or at EL3 when SCTLR_EL3.TME == 0.	ISS - an exception from a TSTART instruction	When FEAT_TME is implemented
0b011100	Exception from a Pointer Authentication instruction authentication failure	ISS - an exception from a Pointer Authentication instruction authentication failure	When FEAT_FPAC is implemented
0b011101	Access to SME functionality trapped as a result of CPACR_EL1.SMEN, CPTR_EL2.SMEN, CPTR_EL2.TSM, CPTR_EL3.ESM, or an attempted execution of an instruction that is illegal because of the value of PSTATE.SM or PSTATE.ZA, that is not reported using EC 0b000000.	ISS - an exception due to SME functionality	When FEAT_SME is implemented
0b011110	Exception from a Granule Protection Check	ISS - an exception from a Granule Protection Check	When FEAT_RME is implemented
0b100000	Instruction Abort from a lower Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from an Instruction Abort	
0b100001	Instruction Abort taken without a change in Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from an Instruction Abort	
0b100010	PC alignment fault exception.	ISS - an exception from an Illegal Execution state, or a PC or SP alignment fault	

Value	Meaning	Link	Applies
`0b100100`	Data Abort from a lower Exception level. Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from a Data Abort	
`0b100101`	Data Abort taken without a change in Exception level. Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from a Data Abort	
`0b100110`	SP alignment fault exception.	ISS - an exception from an Illegal Execution state, or a PC or SP alignment fault	
`0b101000`	Trapped floating-point exception taken from AArch32 state. This EC value is valid if the implementation supports trapping of floating-point exceptions, otherwise it is reserved. Whether a floating-point implementation supports trapping of floating-point exceptions is IMPLEMENTATION DEFINED.	ISS - an exception from a trapped floating-point exception	When AArch32 is supported
`0b101100`	Trapped floating-point exception taken from AArch64 state. This EC value is valid if the implementation supports trapping of floating-point exceptions, otherwise it is reserved. Whether a floating-point implementation supports trapping of floating-point exceptions is IMPLEMENTATION DEFINED.	ISS - an exception from a trapped floating-point exception	When AArch64 is supported
`0b101111`	SError interrupt.	ISS - an SError interrupt	
`0b110000`	Breakpoint exception from a lower Exception level.	ISS - an exception from a Breakpoint or Vector Catch debug exception	
`0b110001`	Breakpoint exception taken without a change in Exception level.	ISS - an exception from a Breakpoint or Vector Catch debug exception	
`0b110010`	Software Step exception from a lower Exception level.	ISS - an exception from a Software Step exception	
`0b110011`	Software Step exception taken without a change in Exception level.	ISS - an exception from a Software Step exception	
`0b110100`	Watchpoint exception from a lower Exception level.	ISS - an exception from a Watchpoint exception	
`0b110101`	Watchpoint exception taken without a change in Exception level.	ISS - an exception from a Watchpoint exception	
`0b111000`	BKPT instruction execution in AArch32 state.	ISS - an exception from execution of a Breakpoint instruction	When AArch32 is supported

Value	Meaning	Link	Applies
0b111100	BRK instruction execution in AArch64 state.	ISS - an exception from execution of a Breakpoint instruction	When AArch64 is supported

All other EC values are reserved by Arm, and:

- Unused values in the range 0b000000 - 0b101100 (0x00 - 0x2C) are reserved for future use for synchronous exceptions.
- Unused values in the range 0b101101 - 0b111111 (0x2D - 0x3F) are reserved for future use, and might be used for synchronous or asynchronous exceptions.

The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [25]

Instruction Length for synchronous exceptions. Possible values of this bit are:

Value	Meaning
0b0	16-bit instruction trapped.
0b1	32-bit instruction trapped. This value is also used when the exception is one of the following: • An SError interrupt. • An Instruction Abort exception. • A PC alignment fault exception. • An SP alignment fault exception. • A Data Abort exception for which the value of the ISV bit is 0. • An Illegal Execution state exception. • Any debug exception except for Breakpoint instruction exceptions. For Breakpoint instruction exceptions, this bit has its standard meaning: – 0b0: 16-bit T32 BKPT instruction. – 0b1: 32-bit A32 BKPT instruction or A64 BRK instruction. • An exception reported using EC value 0b000000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS, bits [24:0]

Instruction Specific Syndrome. Architecturally, this field can be defined independently for each defined Exception class. However, in practice, some ISS encodings are used for more than one Exception class.

Typically, an ISS encoding has a number of subfields. When an ISS subfield holds a register number, the value returned in that field is the AArch64 view of the register number.

For an exception taken from AArch32 state, see 'Mapping of the general-purpose registers between the Execution states'.

If the AArch32 register descriptor is 0b1111, then:

- If the instruction that generated the exception was not UNPREDICTABLE, the field takes the value 0b11111.
- If the instruction that generated the exception was UNPREDICTABLE, the field takes an UNKNOWN value that must be either:
  - The AArch64 view of the register number of a register that might have been used at the Exception level from which the exception was taken.
  - The value 0b11111.

ISS encoding for exceptions with an unknown reason

Bits [24:0]

Reserved, RES0.

Additional information for exceptions with an unknown reason

When an exception is reported using this EC code the IL field is set to 1.

This EC code is used for all exceptions that are not covered by any other EC value. This includes exceptions that are generated in the following situations:

- The attempted execution of an instruction bit pattern that has no allocated instruction or that is not accessible at the current Exception level and Security state, including:
  - A read access using a System register pattern that is not allocated for reads or that does not permit reads at the current Exception level and Security state.
  - A write access using a System register pattern that is not allocated for writes or that does not permit writes at the current Exception level and Security state.
  - Instruction encodings that are unallocated.
  - Instruction encodings for instructions or System registers that are not implemented in the implementation.
- In Debug state, the attempted execution of an instruction bit pattern that is not accessible in Debug state.
- In Non-debug state, the attempted execution of an instruction bit pattern that is not accessible in Non-debug state.
- In AArch32 state, attempted execution of a short vector floating-point instruction.
- In an implementation that does not include Advanced SIMD and floating-point functionality, an attempted access to Advanced SIMD or floating-point functionality under conditions where that access would be permitted if that functionality was present. This includes the attempted execution of an Advanced SIMD or floating-point instruction, and attempted accesses to Advanced SIMD and floating-point System registers.
- An exception generated because of the value of one of the SCTLR_EL1.{ITD, SED, CP15BEN} control bits.
- Attempted execution of:
  - An HVC instruction when disabled by HCR_EL2.HCD or SCR_EL3.HCE.
  - An SMC instruction when disabled by SCR_EL3.SMD.
  - An HLT instruction when disabled by EDSCR.HDE.
- Attempted execution of an MSR or MRS instruction to access SP_EL0 when the value of SPSel.SP is 0.
- Attempted execution of an MSR or MRS instruction using a _EL12 register name when HCR_EL2.E2H == 0.

- Attempted execution, in Debug state, of:
  - A DCPS1 instruction when the value of HCR_EL2.TGE is 1 and EL2 is disabled or not implemented in the current Security state.
  - A DCPS2 instruction from EL1 or EL0 when EL2 is disabled or not implemented in the current Security state.
  - A DCPS3 instruction when the value of EDSCR.SDD is 1, or when EL3 is not implemented.
- When EL3 is using AArch64, attempted execution from Secure EL1 of an SRS instruction using R13_mon. See 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32'.
- In Debug state when the value of EDSCR.SDD is 1, the attempted execution at EL2, EL1, or EL0 of an instruction that is configured to trap to EL3.
- In AArch32 state, the attempted execution of an MRS (banked register) or an MSR (banked register) instruction to SPSR_mon, SP_mon, or LR_mon.
- An exception that is taken to EL2 because the value of HCR_EL2.TGE is 1 that, if the value of HCR_EL2.TGE was 0 would have been reported with an ESR_ELx.EC value of 0b000111.
- In Non-transactional state, attempted execution of a TCOMMIT instruction.

ISS encoding for an exception from a WF instruction*

24	23 20	19 10	9 5	4 3	2	1 0
CV	COND	RES0	RN	RES0	RV	TI

CV, bit [24]

Condition code valid.

Value	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.

- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:10]

Reserved, RES0.

RN, bits [9:5]

When FEAT_WFxT is implemented:

Register Number. Indicates the register number supplied for a WFET or WFIT instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [4:3]

Reserved, RES0.

RV, bit [2]

When FEAT_WFxT is implemented:

Register field Valid.

If TI[1] == 1, then this field indicates whether RN holds a valid register number for the register argument to the trapped WFET or WFIT instruction.

Value	Meaning
`0b0`	Register field invalid.
`0b1`	Register field valid.

If TI[1] == 0, then this field is RES0.

This field is set to 1 on a trap on WFET or WFIT.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TI, bits [1:0]

Trapped instruction. Possible values of this bit are:

Value	Meaning	Applies
`0b00`	WFI trapped.	
`0b01`	WFE trapped.	
`0b10`	WFIT trapped.	When FEAT_WFxT is implemented
`0b11`	WFET trapped.	When FEAT_WFxT is implemented

When FEAT_WFxT is implemented, this is a two bit field as shown. Otherwise, bit[1] is RES0.

The reset behavior of this field is:

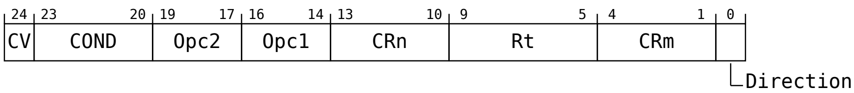
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a WF* instruction

The following fields describe configuration settings for generating this exception:

- SCTLR_EL1.{nTWE, nTWI}.
- HCR_EL2.{TWE, TWI}.
- SCR_EL3.{TWE, TWI}.

ISS encoding for an exception from an MCR or MRC access



CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc2, bits [19:17]

The Opc2 value from the issued instruction.

For a trapped VMRS access, holds the value 0b000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [16:14]

The Opc1 value from the issued instruction.

For a trapped VMRS access, holds the value 0b111.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

For a trapped VMRS access, holds the reg field from the VMRS instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.

- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

For a trapped VMRS access, holds the value 0b0000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
0b0	Write to System register space. MCR instruction.
0b1	Read from System register space. MRC or VMRS instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an MCR or MRC access

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000011:

- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- PMUSERENR_EL0.{ER, CR, SW, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- AMUSERENR_EL0.EN, for accesses to Activity Monitors registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TTLB, for execution of TLB maintenance instructions at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.{TSW, TPC, TPU} for execution of cache maintenance instructions at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TACR, for accesses to the Auxiliary Control Register at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TIDCP, for accesses to lockdown, DMA, and TCM operations at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.{TID1, TID2, TID3}, for accesses to ID registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.

- CPTR_EL2.TCPAC, for accesses to CPACR_EL1 or CPACR using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HSTR_EL2.T<n>, for accesses to System registers using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- CNTHCTL_EL2.EL1PCEN, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- CPTR_EL3.TCPAC, for accesses to CPACR from EL1 and EL2, and accesses to HCPTR from EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- For information on other traps using EC value 0b000011, see ‘Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32’.
- If FEAT_FGT is implemented, MCR or MRC access to some registers at EL0, trapped to EL2.

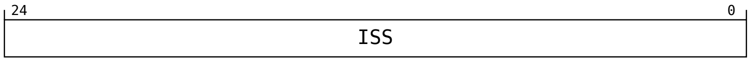
The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000101:

- CPACR_EL1.TTA for accesses to trace registers, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- MDSCR_EL1.TDCC, for accesses to the Debug Communications Channel (DCC) registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- If FEAT_FGT is implemented, MDCR_EL2.TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- HCR_EL2.TID0, for accesses to the JIDR register in the ID group 0 at EL0 and EL1 using AArch32, MRC access (coproc == 0b1110) trapped to EL2.
- CPTR_EL2.TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers DBGDRAR and DBGDSAR using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL2.TDOSA, for accesses to powerdown debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL2.TDA, for accesses to other debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- CPTR_EL3.TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.
- MDCR_EL3.TDOSA, for accesses to powerdown debug registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.
- MDCR_EL3.TDA, for accesses to other debug registers, using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001000:

- HCR_EL2.TID0, for accesses to the FPSID register in ID group 0 at EL1 using AArch32 state, VMRS access trapped to EL2.
- HCR_EL2.TID3, for accesses to registers in ID group 3 including MVFR0, MVFR1 and MVFR2, VMRS access trapped to EL2.

ISS encoding for an exception from an LD64B or ST64B instruction*

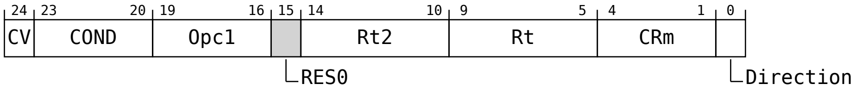


ISS, bits [24:0]

Value	Meaning
`0b0000000000000000000000000`	ST64BV instruction trapped.
`0b0000000000000000000000001`	ST64BV0 instruction trapped.
`0b0000000000000000000000010`	LD64B or ST64B instruction trapped.

All other values are reserved.

ISS encoding for an exception from an MCRR or MRRC access



CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.

- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [19:16]

The Opc1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [15]

Reserved, RES0.

Rt2, bits [14:10]

The Rt2 value from the issued instruction, the second general-purpose register used for the transfer.

If the Rt2 value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt2 value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the first general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.

- The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
`0b0`	Write to System register space. MCRR instruction.
`0b1`	Read from System register space. MRRC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an MCRR or MRRC access

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000100:

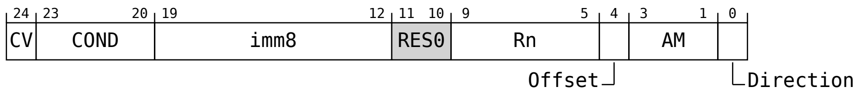
- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- PMUSERENR_EL0.{CR, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- AMUSERENR_EL0.{EN}, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- HSTR_EL2.T<n>, for accesses to System registers using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- CNTHCTL_EL2.{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- If FEAT_FGT is implemented, HDFGRTR_EL2.PMCCNTR_EL0 for MRRC access and HDFGWTR_EL2.PMCCNTR_EL0 for MCRR access to PMCCNTR at EL0, trapped to EL2.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001100:

- MDSCR_EL1.TDCC, for accesses to the Debug ROM registers DBGDSAR and DBGDRAR at EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers DBGDRAR and DBGDSAR using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL3.TDA, for accesses to debug registers, using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.
- CPACR_EL1.TTA for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- CPTR_EL2.TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- CPTR_EL3.TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.

If the Armv8-A architecture is implemented with an ETMv4 implementation, MCRR and MRRC accesses to trace registers are UNDEFINED and the resulting exception is higher priority than an exception due to these traps.

ISS encoding for an exception from an LDC or STC instruction



CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.

- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

imm8, bits [19:12]

The immediate value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:10]

Reserved, RES0.

Rn, bits [9:5]

The Rn value from the issued instruction, the general-purpose register used for the transfer.

If the Rn value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rn value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

This field is valid only when AM[2] is 0, indicating an immediate form of the LDC or STC instruction. When AM[2] is 1, indicating a literal form of the LDC or STC instruction, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Offset, bit [4]

Indicates whether the offset is added or subtracted:

Value	Meaning
`0b0`	Subtract offset.
`0b1`	Add offset.

This bit corresponds to the U bit in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

AM, bits [3:1]

Addressing mode. The permitted values of this field are:

Value	Meaning
`0b000`	Immediate unindexed.
`0b001`	Immediate post-indexed.
`0b010`	Immediate offset.
`0b011`	Immediate pre-indexed.
`0b100`	For a trapped STC instruction or a trapped T32 LDC instruction this encoding is reserved.
`0b110`	For a trapped STC instruction, this encoding is reserved.

The values 0b101 and 0b111 are reserved. The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in System and memory-mapped registers and translation table entries'.

Bit [2] in this subfield indicates the instruction form, immediate or literal.

Bits [1:0] in this subfield correspond to the bits {P, W} in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
`0b0`	Write to memory. STC instruction.
`0b1`	Read from memory. LDC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an LDC or STC instruction

The following fields describe the configuration settings for the traps that are reported using EC value 0b000110:

- MDSCR_EL1.TDCC, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint trapped to EL1 or EL2.
- MDCR_EL2.TDA, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint MCR or MRC access trapped to EL2.
- MDCR_EL3.TDA, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint MCR or MRC access trapped to EL3.

- If FEAT_FGT is implemented, MDCR_EL2.TDCC for LDC and STC accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.

ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps

24	23–20	19–0
CV	COND	RES0

The accesses covered by this trap include:

- Execution of SVE or Advanced SIMD and floating-point instructions.
- Accesses to the Advanced SIMD and floating-point System registers.
- Execution of SME instructions.

For an implementation that does not include either SVE or support for Advanced SIMD and floating-point, the exception is reported using the EC value 0b000000.

CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.

   – CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:0]

Reserved, RES0.

Additional information for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps

The following fields describe the configuration settings for the traps that are reported using EC value 0b000111:

- CPACR_EL1.FPEN, for accesses to SIMD and floating-point registers trapped to EL1.
- CPTR_EL2.FPEN and CPTR_EL2.TFP, for accesses to SIMD and floating-point registers trapped to EL2.
- CPTR_EL3.TFP, for accesses to SIMD and floating-point registers trapped to EL3.

ISS encoding for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ

24 … 0
RES0

The accesses covered by this trap include:

- Execution of SVE instructions when the PE is not in Streaming SVE mode.
- Accesses to the SVE System registers, ZCR_ELx.

For an implementation that does not include SVE, the exception is reported using the EC value 0b000000.

Bits [24:0]

Reserved, RES0.

Additional information for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ

The following fields describe the configuration settings for the traps that are reported using EC value 0b011001:

- CPACR_EL1.ZEN, for execution of SVE instructions and accesses to SVE registers at EL0 or EL1, trapped to EL1.
- CPTR_EL2.ZEN and CPTR_EL2.TZ, for execution of SVE instructions and accesses to SVE registers at EL0, EL1, or EL2, trapped to EL2.
- CPTR_EL3.EZ, for execution of SVE instructions and accesses to SVE registers from all Exception levels, trapped to EL3.

ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault

24 … 0
RES0

Bits [24:0]

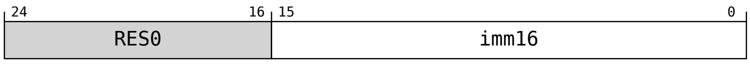
Reserved, RES0.

Additional information for an exception from an Illegal Execution state, or a PC or SP alignment fault

There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about these exceptions, see 'The Illegal Execution state exception' and 'PC alignment checking'.

'SP alignment checking' describes the configuration settings for generating SP alignment fault exceptions.

ISS encoding for an exception from HVC or SVC instruction execution



Bits [24:16]

Reserved, RES0.

imm16, bits [15:0]

The value of the immediate field from the HVC or SVC instruction.

For an HVC instruction, and for an A64 SVC instruction, this is the value of the imm16 field of the issued instruction.

For an A32 or T32 SVC instruction:

- If the instruction is unconditional, then:
  - For the T32 instruction, this field is zero-extended from the imm8 field of the instruction.
  - For the A32 instruction, this field is the bottom 16 bits of the imm24 field of the instruction.
- If the instruction is conditional, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from HVC or SVC instruction execution

In AArch32 state, the HVC instruction is unconditional, and a conditional SVC instruction generates an exception only if it passes its condition code check. Therefore, the syndrome information for these exceptions does not require conditionality information.

For T32 and A32 instructions, see 'SVC' and 'HVC'.

For A64 instructions, see 'SVC' and 'HVC'.

If FEAT_FGT is implemented, HFGITR_EL2.{SVC_EL1, SVC_EL0} control fine-grained traps on SVC execution.

ISS encoding for an exception from SMC instruction execution in AArch32 state

For an SMC instruction that completes normally and generates an exception that is taken to EL3, the ISS encoding is RES0.

For an SMC instruction that is trapped to EL2 from EL1 because HCR_EL2.TSC is 1, the ISS encoding is as shown in the diagram.

CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

This field is valid only if CCKNOWNPASS is 1, otherwise it is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

This field is valid only if CCKNOWNPASS is 1, otherwise it is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CCKNOWNPASS, bit [19]

Indicates whether the instruction might have failed its condition code check.

Value	Meaning
`0b0`	The instruction was unconditional, or was conditional and passed its condition code check.
`0b1`	The instruction was conditional, and might have failed its condition code check.

In an implementation in which an SMC instruction that fails it code check is not trapped, this field can always return the value 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [18:0]

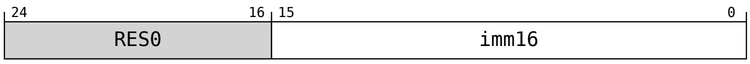
Reserved, RES0.

Additional information for an exception from SMC instruction execution in AArch32 state

HCR_EL2.TSC describes the configuration settings for trapping SMC instructions to EL2.

'System calls' describes the case where these exceptions are trapped to EL3.

ISS encoding for an exception from SMC instruction execution in AArch64 state



Bits [24:16]

Reserved, RES0.

imm16, bits [15:0]

The value of the immediate field from the issued SMC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from SMC instruction execution in AArch64 state

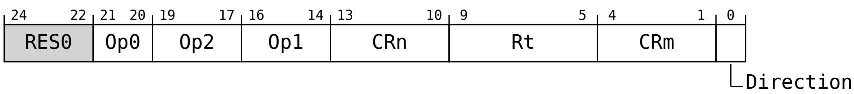
The value of ISS[24:0] described here is used both:

- When an SMC instruction is trapped from EL1 modes.
- When an SMC instruction is not trapped, so completes normally and generates an exception that is taken to EL3.

HCR_EL2.TSC describes the configuration settings for trapping SMC from EL1 modes.

'System calls' describes the case where these exceptions are trapped to EL3.

ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state



Bits [24:22]

Reserved, RES0.

Op0, bits [21:20]

The Op0 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op2, bits [19:17]

The Op2 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op1, bits [16:14]

The Op1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
`0b0`	Write access, including MSR instructions.
`0b1`	Read access, including MRS instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from MSR, MRS, or System instruction execution in AArch64 state

For exceptions caused by System instructions, see 'System instructions' subsection of 'Branches, exception generating and System instructions' for the encoding values returned by an instruction.

The following fields describe configuration settings for generating the exception that is reported using EC value 0b011000:

- SCTLR_EL1.UCI, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- SCTLR_EL1.UCT, for accesses to CTR_EL0 using AArch64 state, MSR or MRS access trapped to EL1 or EL2.

- SCTLR_EL1.DZE, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- SCTLR_EL1.UMA, for accesses to the PSTATE interrupt masks using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- CPACR_EL1.TTA, for accesses to the trace registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- MDSCR_EL1.TDCC, for accesses to the Debug Communications Channel (DCC) registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- If FEAT_FGT is implemented, MDCR_EL2.TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN} accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- PMUSERENR_EL0.{ER, CR, SW, EN}, for accesses to the Performance Monitor registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- AMUSERENR_EL0.EN, for accesses to Activity Monitors registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TDZ, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TTLB, for execution of TLB maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{TSW, TPC, TPU}, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TACR, for accesses to the Auxiliary Control Register, ACTLR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TIDCP, for accesses to lockdown, DMA, and TCM operations using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{TID1, TID2, TID3}, for accesses to ID group 1, ID group 2 or ID group 3 registers, using AArch64 state, MSR or MRS access trapped to EL2.
- CPTR_EL2.TCPAC, for accesses to CPACR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.
- CPTR_EL2.TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TTRF, for accesses to the trace filter control register, TRFCR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDOSA, for accesses to powerdown debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- CNTHCTL_EL2.{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDA, for accesses to debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.APK, for accesses to Pointer authentication key registers. using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{NV, NV1}, for Nested virtualization register access, using AArch64 state, MSR or MRS access, trapped to EL2.
- HCR_EL2.AT, for execution of AT S1E* instructions, using AArch64 state, MSR or MRS access, trapped to EL2.
- HCR_EL2.{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access, trapped to EL2.

- SCR_EL3.APK, for accesses to Pointer authentication key registers, using AArch64 state, MSR or MRS access trapped to EL3.
- SCR_EL3.ST, for accesses to the Counter-timer Physical Secure timer registers, using AArch64 state, MSR or MRS access trapped to EL3.
- SCR_EL3.{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TCPAC, for accesses to CPTR_EL2 and CPACR_EL1 using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TTRF, for accesses to the trace filter control registers, TRFCR_EL1 and TRFCR_EL2, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TDA, for accesses to debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TDOSA, for accesses to powerdown debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access, trapped to EL3.
- If FEAT_EVT is implemented, the following registers control traps for EL1 and EL0 Cache controls that use this EC value:
  - HCR_EL2.{TTLBOS, TTLBIS, TICAB, TOCU, TID4}.
  - HCR2.{TTLBIS, TICAB, TOCU, TID4}.
- If FEAT_FGT is implemented:
  - SCR_EL3.FGTEn, for accesses to the fine-grained trap registers, MSR or MRS access at EL2 trapped to EL3.
  - HFGRTR_EL2 for reads and HFGWTR_EL2 for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 trapped to EL2.
  - HFGITR_EL2 for execution of system instructions, MSR or MRS access trapped to EL2
  - HDFGRTR_EL2 for reads and HDFGWTR_EL2 for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 state trapped to EL2.
  - HAFGRTR_EL2 for reads of Activity Monitor counters, using AArch64 state, MRS access at EL0 and EL1 trapped to EL2.
- If FEAT_RNG_TRAP is implemented:
  - SCR_EL3.TRNDR for reads of RNDR and RNDRRS using AArch64 state, MRS access trapped to EL3.
- If FEAT_SME is implemented:
  - CPTR_EL3.ESM, for MSR or MRS accesses to SMPRI_EL1 at EL1, EL2, and EL3, trapped to EL3.
  - CPTR_EL3.ESM, for MSR or MRS accesses to SMPRIMAP_EL2 at EL2 and EL3, trapped to EL3.
  - SCTLR_EL1.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, trapped to EL1 or EL2.
  - SCTLR_EL2.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, trapped to EL2.
  - SCR_EL3.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, EL1, and EL2, trapped to EL3.

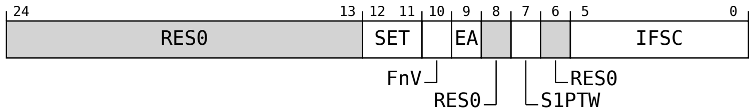
ISS encoding for an IMPLEMENTATION DEFINED exception to EL3

24 … 0
IMPLEMENTATION DEFINED

IMPLEMENTATION DEFINED, bits [24:0]

IMPLEMENTATION DEFINED

ISS encoding for an exception from an Instruction Abort



Bits [24:13]

Reserved, RES0.

SET, bits [12:11]

When FEAT_RAS is implemented:

Synchronous Error Type. When IFSC is 0b010000, describes the PE error state after taking the Instruction Abort exception.

Value	Meaning
0b00	Recoverable state (UER).
0b10	Uncontainable (UC).
0b11	Restartable state (UEO).

All other values are reserved.

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

Value	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [8]

Reserved, RES0.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

Value	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

Value	Meaning	Applies
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented

Value	Meaning	Applies
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented

Value	Meaning	Applies
`0b100111`	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
`0b101000`	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
`0b101001`	Address size fault, level -1.	When FEAT_LPA2 is implemented
`0b101011`	Translation fault, level -1.	When FEAT_LPA2 is implemented
`0b110000`	TLB conflict abort.	
`0b110001`	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

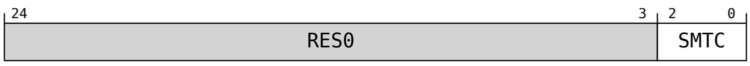
Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception due to SME functionality



The accesses covered by this trap include:

- Execution of SME instructions.
- Execution of SVE and Advanced SIMD instructions, when the PE is in Streaming SVE mode.
- Direct accesses of SVCR, SMCR_EL1, SMCR_EL2, SMCR_EL3.

Bits [24:3]

Reserved, RES0.

SMTC, bits [2:0]

SME Trap Code. Identifies the reason for instruction trapping.

Value	Meaning
`0b000`	Access to SME functionality trapped as a result of CPACR_EL1.SMEN, CPTR_EL2.SMEN, CPTR_EL2.TSM, or CPTR_EL3.ESM, that is not reported using EC 0b000000.

Value	Meaning
0b001	Advanced SIMD, SVE, or SVE2 instruction trapped because PSTATE.SM is 1.
0b010	SME instruction trapped because PSTATE.SM is 0.
0b011	SME instruction trapped because PSTATE.ZA is 0.

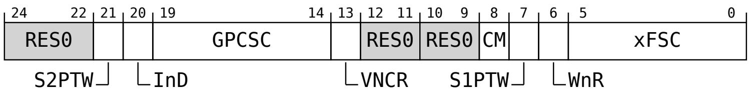
All other values are reserved.

Additional information for an exception due to SME functionality

The following fields describe the configuration settings for the traps that are reported using the EC value 0b011101:

- CPACR_EL1.SMEN, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR and SMCR_EL1 System registers at EL1 and EL0, trapped to EL1 or EL2.
- CPTR_EL2.SMEN and CPTR_EL2.TSM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR, SMCR_EL1, SMCR_EL2 at EL2, EL1, or EL0, trapped to EL2.
- CPTR_EL3.ESM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR, SMCR_EL1, SMCR_EL2, SMCR_EL3 from all Exception levels and any Security state, trapped to EL3.

ISS encoding for an exception from a Granule Protection Check



Bits [24:22]

Reserved, RES0.

S2PTW, bit [21]

Indicates whether the Granule Protection Check exception was on an access made for a stage 2 translation table walk.

Value	Meaning
0b0	Fault not on a stage 2 translation table walk.
0b1	Fault on a stage 2 translation table walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

InD, bit [20]

Indicates whether the Granule Protection Check exception was on an instruction or data access.

Value	Meaning
0b0	Data access.

Value	Meaning
0b1	Instruction access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GPCSC, bits [19:14]

Granule Protection Check Status Code.

Value	Meaning
0b000000	GPT address size fault at level 0.
0b000001	GPT address size fault at level 1.
0b000100	GPT walk fault at level 0.
0b000101	GPT walk fault at level 1.
0b001100	Granule protection fault at level 0.
0b001101	Granule protection fault at level 1.
0b010100	Synchronous External abort on GPT fetch at level 0.
0b010101	Synchronous External abort on GPT fetch at level 1.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [13]

When FEAT_NV2 is implemented

VNCR, bit [13]

Indicates that the fault came from use of VNCR_EL2 register by EL1 code.

Value	Meaning
0b0	The fault was not generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.
0b1	The fault was generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

When InD is '1', this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

Bit [13]

Reserved, RES0.

Bits [12:11]

Reserved, RES0.

Bits [10:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

Value	Meaning
`0b0`	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
`0b1`	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

Indicates whether the Granule Protection Check exception was on an access for stage 2 translation for a stage 1 translation table walk:

Value	Meaning
`0b0`	Fault not on a stage 2 translation for a stage 1 translation table walk.
`0b1`	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

Value	Meaning
`0b0`	Abort caused by an instruction reading from a memory location.
`0b1`	Abort caused by an instruction writing to a memory location.

When InD is '1', this field is RES0.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

xFSC, bits [5:0]

Instruction or Data Fault Status Code.

Value	Meaning	Applies
`0b100011`	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
`0b100100`	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
`0b100101`	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
`0b100110`	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
`0b100111`	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
`0b101000`	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

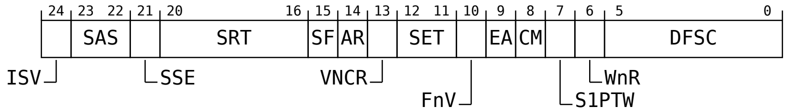
Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a Data Abort



When FEAT_LS64 is implemented, if a memory access generated by an ST64BV or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this ISS encoding includes ISS2, bits[36:32].

ISV, bit [24]

Instruction Syndrome Valid. Indicates whether the syndrome information in ISS[23:14] is valid.

Value	Meaning
0b0	No valid instruction syndrome. ISS[23:14] are RES0.
0b1	ISS[23:14] hold a valid instruction syndrome.

In ESR_EL2, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For other faults reported in ESR_EL2, ISV is 0 except for the following stage 2 aborts:

- AArch64 loads and stores of a single general-purpose register (including the register specified with 0b11111, including those with Acquire/Release semantics, but excluding Load Exclusive or Store Exclusive and excluding those with writeback).
- AArch32 instructions where the instruction:
  - Is an LDR, LDA, LDRT, LDRSH, LDRSHT, LDRH, LDAH, LDRHT, LDRSB, LDRSBT, LDRB, LDAB, LDRBT, STR, STL, STRT, STRH, STLH, STRHT, STRB, STLB, or STRBT instruction.
  - Is not performing register writeback.
  - Is not using R15 as a source or destination register.

For these stage 2 aborts, ISV is UNKNOWN if the exception was generated in Debug state in memory access mode, and otherwise indicates whether ISS[23:14] hold a valid syndrome.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault. ISV is 0 for all other faults reported in ESR_EL1 or ESR_EL3.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

For ISS reporting, a stage 2 abort on a stage 1 translation table walk does not return a valid instruction syndrome, and therefore ISV is 0 for these aborts.

When FEAT_RAS is not implemented, it is IMPLEMENTATION DEFINED whether ISV is set to 1 or 0 on a synchronous External abort on a stage 2 translation table walk.

When FEAT_MTE2 is implemented, for a synchronous Tag Check Fault abort taken to ELx, ESR_ELx.FNV is 0 and FAR_ELx is valid.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SAS, bits [23:22]

When ISV == 1:

Syndrome Access Size. Indicates the size of the access attempted by the faulting operation.

Value	Meaning
0b00	Byte
0b01	Halfword
0b10	Word
0b11	Doubleword

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SSE, bit [21]

When ISV == 1:

Syndrome Sign Extend. For a byte, halfword, or word load operation, indicates whether the data item must be sign extended.

Value	Meaning
0b0	Sign-extension not required.
0b1	Data item must be sign-extended.

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

For all other operations, this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SRT, bits [20:16]

When ISV == 1:

Syndrome Register Transfer. The register number of the Wt/Xt/Rt operand of the faulting instruction.

If the exception was taken from an Exception level that is using AArch32, then this is the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states'.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SF, bit [15]

When ISV == 1:

Width of the register accessed by the instruction is Sixty-Four.

Value	Meaning
0b0	Instruction loads/stores a 32-bit wide register.
0b1	Instruction loads/stores a 64-bit wide register.

This field specifies the register width identified by the instruction, not the Execution state.

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

AR, bit [14]

When ISV == 1:

Acquire/Release.

Value	Meaning
0b0	Instruction did not have acquire/release semantics.
0b1	Instruction did have acquire/release semantics.

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bit [13]

When FEAT_NV2 is implemented

VNCR, bit [13]

Indicates that the fault came from use of VNCR_EL2 register by EL1 code.

Value	Meaning
`0b0`	The fault was not generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.
`0b1`	The fault was generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

Bit [13]

Reserved, RES0.

Bits [12:11]

When FEAT_RAS is implemented and FEAT_LS64 is not implemented

SET, bits [12:11]

Synchronous Error Type. When DFSC is 0b010000, describes the PE error state after taking the Data Abort exception.

Value	Meaning
`0b00`	Recoverable state (UER).
`0b10`	Uncontainable (UC).
`0b11`	Restartable state (UEO).

All other values are reserved.

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

This field is valid only if the DFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_LS64 is implemented

LST, bits [12:11]

Load/Store Type. Used when an LD64B, ST64B, ST64BV, or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

Value	Meaning
`0b01`	An ST64BV instruction generated the Data Abort.
`0b10`	An LD64B or ST64B instruction generated the Data Abort.
`0b11`	An ST64BV0 instruction generated the Data Abort.

All other values are reserved.

This field is valid only if the DFSC code is 0b110101. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

Value	Meaning
`0b0`	FAR is valid.
`0b1`	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the DFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

Value	Meaning
`0b0`	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.

Value	Meaning
`0b1`	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

Value	Meaning
`0b0`	Fault not on a stage 2 translation for a stage 1 translation table walk.
`0b1`	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

Value	Meaning
`0b0`	Abort caused by an instruction reading from a memory location.
`0b1`	Abort caused by an instruction writing to a memory location.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DFSC, bits [5:0]

Data Fault Status Code.

Value	Meaning	Applies
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Synchronous Tag Check Fault.	When FEAT_MTE2 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented

Value	Meaning	Applies
`0b011011`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
`0b011100`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
`0b011101`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
`0b011110`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
`0b011111`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
`0b100001`	Alignment fault.	
`0b100011`	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
`0b100100`	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
`0b100101`	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
`0b100110`	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
`0b100111`	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
`0b101000`	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
`0b101001`	Address size fault, level -1.	When FEAT_LPA2 is implemented
`0b101011`	Translation fault, level -1.	When FEAT_LPA2 is implemented
`0b110000`	TLB conflict abort.	
`0b110001`	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
`0b110100`	IMPLEMENTATION DEFINED fault (Lockdown).	
`0b110101`	IMPLEMENTATION DEFINED fault (Unsupported Exclusive or Atomic access).	

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

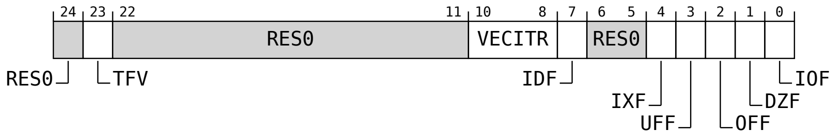
Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a trapped floating-point exception



Bit [24]

Reserved, RES0.

TFV, bit [23]

Trapped Fault Valid bit. Indicates whether the IDF, IXF, UFF, OFF, DZF, and IOF bits hold valid information about trapped floating-point exceptions.

Value	Meaning
0b0	The IDF, IXF, UFF, OFF, DZF, and IOF bits do not hold valid information about trapped floating-point exceptions and are UNKNOWN.
0b1	One or more floating-point exceptions occurred during an operation performed while executing the reported instruction. The IDF, IXF, UFF, OFF, DZF, and IOF bits indicate trapped floating-point exceptions that occurred. For more information, see 'Floating-point exceptions and exception traps'.

It is IMPLEMENTATION DEFINED whether this field is set to 0 on an exception generated by a trapped floating-point exception from an instruction that is performing floating-point operations on more than one lane of a vector.

This is not a requirement. Implementations can set this field to 1 on a trapped floating-point exception from an instruction and return valid information in the {IDF, IXF, UFF, OFF, DZF, IOF} fields.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [22:11]

Reserved, RES0.

VECITR, bits [10:8]

For a trapped floating-point exception from an instruction executed in AArch32 state this field is RES1.

For a trapped floating-point exception from an instruction executed in AArch64 state this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IDF, bit [7]

Input Denormal floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
0b0	Input denormal floating-point exception has not occurred.
0b1	Input denormal floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

IXF, bit [4]

Inexact floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
0b0	Inexact floating-point exception has not occurred.
0b1	Inexact floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

UFF, bit [3]

Underflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
0b0	Underflow floating-point exception has not occurred.
0b1	Underflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

OFF, bit [2]

Overflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Overflow floating-point exception has not occurred.
`0b1`	Overflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DZF, bit [1]

Divide by Zero floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Divide by Zero floating-point exception has not occurred.
`0b1`	Divide by Zero floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IOF, bit [0]

Invalid Operation floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Invalid Operation floating-point exception has not occurred.
`0b1`	Invalid Operation floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

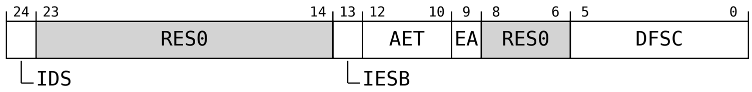
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a trapped floating-point exception

In an implementation that supports the trapping of floating-point exceptions:

- From an Exception level using AArch64, the FPCR.{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.
- From an Exception level using AArch32, the FPSCR.{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.

ISS encoding for an SError interrupt



IDS, bit [24]

IMPLEMENTATION DEFINED syndrome.

Value	Meaning
0b0	Bits [23:0] of the ISS field holds the fields described in this encoding. If FEAT_RAS is not implemented, bits [23:0] of the ISS field are RES0.
0b1	Bits [23:0] of the ISS field holds IMPLEMENTATION DEFINED syndrome information that can be used to provide additional information about the SError interrupt.

This field was previously called ISV.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [23:14]

Reserved, RES0.

IESB, bit [13]

When FEAT_IESB is implemented:

Implicit error synchronization event.

Value	Meaning
0b0	The SError interrupt was either not synchronized by the implicit error synchronization event or not taken immediately.
0b1	The SError interrupt was synchronized by the implicit error synchronization event and taken immediately.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

AET, bits [12:10]

When FEAT_RAS is implemented:

Asynchronous Error Type.

When DFSC is 0b010001, describes the PE error state after taking the SError interrupt exception.

Value	Meaning
`0b000`	Uncontainable (UC).
`0b001`	Unrecoverable state (UEU).
`0b010`	Restartable state (UEO).
`0b011`	Recoverable state (UER).
`0b110`	Corrected (CE).

All other values are reserved.

If multiple errors are taken as a single SError interrupt exception, the overall PE error state is reported.

Software can use this information to determine what recovery might be possible. The recovery software must also examine any implemented fault records to determine the location and extent of the error.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EA, bit [9]

When FEAT_RAS is implemented:

External abort type. When DFSC is 0b010001, provides an IMPLEMENTATION DEFINED classification of External aborts.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [8:6]

Reserved, RES0.

DFSC, bits [5:0]

When FEAT_RAS is implemented:

Data Fault Status Code.

Value	Meaning
`0b000000`	Uncategorized error.
`0b010001`	Asynchronous SError interrupt.

All other values are reserved.

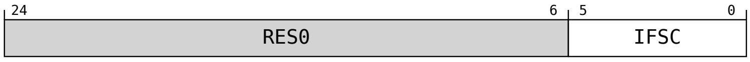
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

ISS encoding for an exception from a Breakpoint or Vector Catch debug exception



Bits [24:6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

Value	Meaning
0b100010	Debug exception.

The reset behavior of this field is:

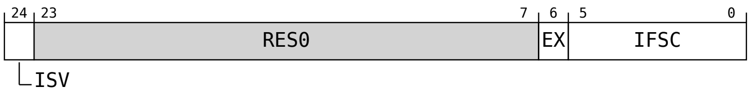
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a Breakpoint or Vector Catch debug exception

For more information about generating these exceptions:

- For exceptions from AArch64, see 'Breakpoint exceptions'.
- For exceptions from AArch32, see 'Breakpoint exceptions' and 'Vector Catch exceptions'.

ISS encoding for an exception from a Software Step exception



ISV, bit [24]

Instruction syndrome valid. Indicates whether the EX bit, ISS[6], is valid, as follows:

Value	Meaning
0b0	EX bit is RES0.
0b1	EX bit is valid.

See the EX bit description for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [23:7]

Reserved, RES0.

EX, bit [6]

Exclusive operation. If the ISV bit is set to 1, this bit indicates whether a Load-Exclusive instruction was stepped.

Value	Meaning
`0b0`	An instruction other than a Load-Exclusive instruction was stepped.
`0b1`	A Load-Exclusive instruction was stepped.

If the ISV bit is set to 0, this bit is RES0, indicating no syndrome data is available.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IFSC, bits [5:0]

Instruction Fault Status Code.

Value	Meaning
`0b100010`	Debug exception.

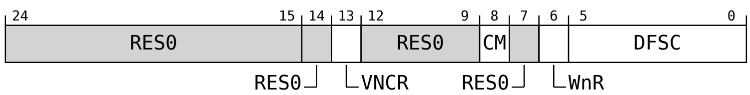
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a Software Step exception

For more information about generating these exceptions, see ‘Software Step exceptions’.

ISS encoding for an exception from a Watchpoint exception



Bits [24:15]

Reserved, RES0.

Bit [14]

Reserved, RES0.

Bit [13]

When FEAT_NV2 is implemented

VNCR, bit [13]

Indicates that the watchpoint came from use of VNCR_EL2 register by EL1 code.

Value	Meaning
`0b0`	The watchpoint was not generated by the use of VNCR_EL2 by EL1 code.
`0b1`	The watchpoint was generated by the use of VNCR_EL2 by EL1 code.

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

Bit [13]

Reserved, RES0.

Bits [12:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Watchpoint exception came from a cache maintenance or address translation instruction:

Value	Meaning
`0b0`	The Watchpoint exception was not generated by the execution of one of the System instructions identified in the description of value 1.
`0b1`	The Watchpoint exception was generated by either the execution of a cache maintenance instruction or by a synchronous Watchpoint exception on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as a cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [7]

Reserved, RES0.

WnR, bit [6]

Write not Read. Indicates whether the Watchpoint exception was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

Value	Meaning
`0b0`	Watchpoint exception caused by an instruction reading from a memory location.

Value	Meaning
`0b1`	Watchpoint exception caused by an instruction writing to a memory location.

For Watchpoint exceptions on cache maintenance and address translation instructions, this bit always returns a value of 1.

For Watchpoint exceptions from an atomic instruction, this field is set to 0 if a read of the location would have generated the Watchpoint exception, otherwise it is set to 1.

If multiple watchpoints match on the same access, it is UNPREDICTABLE which watchpoint generates the Watchpoint exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DFSC, bits [5:0]

Data Fault Status Code.

Value	Meaning
`0b100010`	Debug exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a Watchpoint exception

For more information about generating these exceptions, see 'Watchpoint exceptions'.

ISS encoding for an exception from execution of a Breakpoint instruction

24 – 16	15 – 0
RES0	Comment

Bits [24:16]

Reserved, RES0.

Comment, bits [15:0]

Set to the instruction comment field value, zero extended as necessary.

For the AArch32 BKPT instructions, the comment field is described as the immediate field.

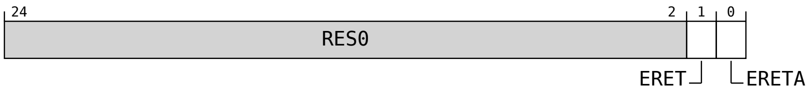
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from execution of a Breakpoint instruction

For more information about generating these exceptions, see 'Breakpoint instruction exceptions'.

ISS encoding for an exception from an ERET, ERETAA, or ERETAB instruction



This EC value applies when FEAT_FGT is implemented, or when HCR_EL2.NV is 1.

Bits [24:2]

Reserved, RES0.

ERET, bit [1]

Indicates whether an ERET or ERETA* instruction was trapped to EL2.

Value	Meaning
0b0	ERET instruction trapped to EL2.
0b1	ERETAA or ERETAB instruction trapped to EL2.

If this bit is 0, the ERETA field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ERETA, bit [0]

Indicates whether an ERETAA or ERETAB instruction was trapped to EL2.

Value	Meaning
0b0	ERETAA instruction trapped to EL2.
0b1	ERETAB instruction trapped to EL2.

When the ERET field is 0, this bit is RES0.

The reset behavior of this field is:

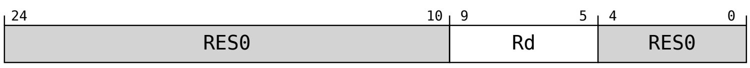
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an ERET, ERETAA, or ERETAB instruction

For more information about generating these exceptions, see HCR_EL2.NV.

If FEAT_FGT is implemented, HFGITR_EL2.ERET controls fine-grained trap exceptions from ERET, ERETAA and ERETAB execution.

ISS encoding for an exception from a TSTART instruction



Bits [24:10]

Reserved, RES0.

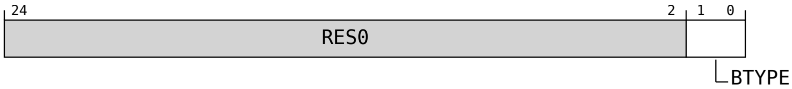
Rd, bits [9:5]

The Rd value from the issued instruction, the general purpose register used for the destination.

Bits [4:0]

Reserved, RES0.

ISS encoding for an exception from Branch Target Identification instruction



Bits [24:2]

Reserved, RES0.

BTYPE, bits [1:0]

This field is set to the PSTATE.BTYPE value that generated the Branch Target Exception.

Additional information for an exception from Branch Target Identification instruction

For more information about generating these exceptions, see 'The AArch64 application level programmers model'.

ISS encoding for an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 || SCR_EL3.API == 0

Bits [24:0]

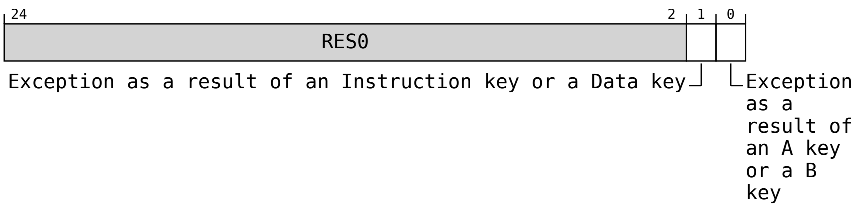
Reserved, RES0.

Additional information for an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 || SCR_EL3.API == 0

For more information about generating these exceptions, see:

- HCR_EL2.API, for exceptions from Pointer authentication instructions, using AArch64 state, trapped to EL2.
- SCR_EL3.API, for exceptions from Pointer authentication instructions, using AArch64 state, trapped to EL3.

ISS encoding for an exception from a Pointer Authentication instruction authentication failure



Bits [24:2]

Reserved, RES0.

Bit [1]

This field indicates whether the exception is as a result of an Instruction key or a Data key.

Value	Meaning
`0b0`	Instruction Key.
`0b1`	Data Key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [0]

This field indicates whether the exception is as a result of an A key or a B key.

Value	Meaning
`0b0`	A key.
`0b1`	B key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a Pointer Authentication instruction authentication failure

The following instructions generate an exception when the Pointer Authentication Code (PAC) is incorrect:

- AUTIASP, AUTIAZ, AUTIA1716.
- AUTIBSP, AUTIBZ, AUTIB1716.
- AUTIA, AUTDA, AUTIB, AUTDB.
- AUTIZA, AUTIZB, AUTDZA, AUTDZB.

It is IMPLEMENTATION DEFINED whether the following instructions generate an exception directly from the authorization failure, rather than changing the address in a way that will generate a Translation fault when the address is accessed:

- RETAA, RETAB.
- BRAA, BRAB, BLRAA, BLRAB.
- BRAAZ, BRABZ, BLRAAZ, BLRABZ.
- ERETAA, ERETAB.
- LDRAA, LDRAB, whether the authenticated address is written back to the base register or not.

Accessing the ESR_EL1

When HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic ESR_EL1 or ESR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, ESR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGRTR_EL2.ESR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x138];
    else
        return ESR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return ESR_EL2;
    else
        return ESR_EL1;
elsif PSTATE.EL == EL3 then
    return ESR_EL1;
```

MSR ESR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGWTR_EL2.ESR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x138] = X[t];
    else
        ESR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        ESR_EL2 = X[t];
    else
        ESR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ESR_EL1 = X[t];
```

MRS <Xt>, ESR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        return NVMem[0x138];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return ESR_EL1;
    else
        UNDEFINED;
```

```
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        return ESR_EL1;
    else
        UNDEFINED;
```

MSR ESR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x138] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        ESR_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        ESR_EL1 = X[t];
    else
        UNDEFINED;
```

MRS <Xt>, ESR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return ESR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return ESR_EL2;
elsif PSTATE.EL == EL3 then
    return ESR_EL2;
```

MSR ESR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        ESR_EL1 = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    ESR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    ESR_EL2 = X[t];
```

A2.1.6 ESR_EL2, Exception Syndrome Register (EL2)

The ESR_EL2 characteristics are:

Purpose

Holds syndrome information for an exception taken to EL2.

Attributes

ESR_EL2 is a 64-bit register.

Configuration

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

AArch64 system register ESR_EL2 bits [31:0] are architecturally mapped to AArch32 system register HSR[31:0].

Field descriptions

The ESR_EL2 bit assignments are:

63–37	36–32
RES0	ISS2

31–26	25	24–0
EC	IL	ISS

ESR_EL2 is made UNKNOWN as a result of an exception return from EL2.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to EL2, the value of ESR_EL2 is UNKNOWN. The value written to ESR_EL2 must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not UNPREDICTABLE at that Exception level, in order to avoid the possibility of a privilege violation.

Bits [63:37]

Reserved, RES0.

ISS2, bits [36:32]

When FEAT_LS64 is implemented:

If a memory access generated by an ST64BV or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

For any other Data Abort, this field is RES0.

Otherwise:

RES0

EC, bits [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about.

For each EC value, the table references a subsection that gives information about:

- The cause of the exception, for example the configuration required to enable the trap.
- The encoding of the associated ISS.

Possible values of the EC field are:

Value	Meaning	Link	Applies
0b000000	Unknown reason.	ISS - exceptions with an unknown reason	
0b000001	Trapped WF* instruction execution. Conditional WF* instructions that fail their condition code check do not cause an exception.	ISS - an exception from a WF* instruction	
0b000011	Trapped MCR or MRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS - an exception from an MCR or MRC access	When AArch32 is supported
0b000100	Trapped MCRR or MRRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS - an exception from an MCRR or MRRC access	When AArch32 is supported
0b000101	Trapped MCR or MRC access with (coproc==0b1110).	ISS - an exception from an MCR or MRC access	When AArch32 is supported
0b000110	Trapped LDC or STC access. The only architected uses of these instruction are: • An STC to write data to memory from DBGDTRRXint. • An LDC to read data from memory to DBGDTRTXint.	ISS - an exception from an LDC or STC instruction	When AArch32 is supported
0b000111	Access to SME, SVE, Advanced SIMD or floating-point functionality trapped by CPACR_EL1.FPEN, CPTR_EL2.FPEN, CPTR_EL2.TFP, or CPTR_EL3.TFP control. Excludes exceptions resulting from CPACR_EL1 when the value of HCR_EL2.TGE is 1, or because SVE or Advanced SIMD and floating-point are not implemented. These are reported with EC value 0b000000 as described in 'The EC used to report an exception routed to EL2 because HCR_EL2.TGE is 1'.	ISS - an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps	
0b001000	Trapped VMRS access, from ID group trap, that is not reported using EC 0b000111.	ISS - an exception from an MCR or MRC access	When AArch32 is supported
0b001001	Trapped use of a Pointer authentication instruction because HCR_EL2.API == 0 || SCR_EL3.API == 0.	ISS - an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 || SCR_EL3.API == 0	When FEAT_PAuth is implemented
0b001010	Trapped execution of an LD64B, ST64B, ST64BV, or ST64BV0 instruction.	ISS - an exception from an LD64B or ST64B* instruction	When FEAT_LS64 is implemented
0b001100	Trapped MRRC access with (coproc==0b1110).	ISS - an exception from an MCRR or MRRC access	When AArch32 is supported
0b001101	Branch Target Exception.	ISS - an exception from Branch Target Identification instruction	When FEAT_BTI is implemented
0b001110	Illegal Execution state.	ISS - an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b010001	SVC instruction execution in AArch32 state. This is reported in ESR_EL2 only when the exception is generated because the value of HCR_EL2.TGE is 1.	ISS - an exception from HVC or SVC instruction execution	When AArch32 is supported

Value	Meaning	Link	Applies
`0b010010`	HVC instruction execution in AArch32 state, when HVC is not disabled.	ISS - an exception from HVC or SVC instruction execution	When AArch32 is supported
`0b010011`	SMC instruction execution in AArch32 state, when SMC is not disabled. This is reported in ESR_EL2 only when the exception is generated because the value of HCR_EL2.TSC is 1.	ISS - an exception from SMC instruction execution in AArch32 state	When AArch32 is supported
`0b010101`	SVC instruction execution in AArch64 state.	ISS - an exception from HVC or SVC instruction execution	When AArch64 is supported
`0b010110`	HVC instruction execution in AArch64 state, when HVC is not disabled.	ISS - an exception from HVC or SVC instruction execution	When AArch64 is supported
`0b010111`	SMC instruction execution in AArch64 state, when SMC is not disabled. This is reported in ESR_EL2 only when the exception is generated because the value of HCR_EL2.TSC is 1.	ISS - an exception from SMC instruction execution in AArch64 state	When AArch64 is supported
`0b011000`	Trapped MSR, MRS or System instruction execution in AArch64 state, that is not reported using EC 0b000000, 0b000001 or 0b000111. This includes all instructions that cause exceptions that are part of the encoding space defined in 'System instruction class encoding overview', except for those exceptions reported using EC values 0b000000, 0b000001, or 0b000111.	ISS - an exception from MSR, MRS, or System instruction execution in AArch64 state	When AArch64 is supported
`0b011001`	Access to SVE functionality trapped as a result of CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ, that is not reported using EC 0b000000.	ISS - an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ	When FEAT_SVE is implemented
`0b011010`	Trapped ERET, ERETAA, or ERETAB instruction execution.	ISS - an exception from an ERET, ERETAA, or ERETAB instruction	When FEAT_PAuth is implemented and FEAT_NV is implemented
`0b011011`	Exception from an access to a TSTART instruction at EL0 when SCTLR_EL1.TME0 == 0, EL0 when SCTLR_EL2.TME0 == 0, at EL1 when SCTLR_EL1.TME == 0, at EL2 when SCTLR_EL2.TME == 0 or at EL3 when SCTLR_EL3.TME == 0.	ISS - an exception from a TSTART instruction	When FEAT_TME is implemented
`0b011100`	Exception from a Pointer Authentication instruction authentication failure	ISS - an exception from a Pointer Authentication instruction authentication failure	When FEAT_FPAC is implemented
`0b011101`	Access to SME functionality trapped as a result of CPACR_EL1.SMEN, CPTR_EL2.SMEN, CPTR_EL2.TSM, CPTR_EL3.ESM, or an attempted execution of an instruction that is illegal because of the value of PSTATE.SM or PSTATE.ZA, that is not reported using EC 0b000000.	ISS - an exception due to SME functionality	When FEAT_SME is implemented
`0b011110`	Exception from a Granule Protection Check	ISS - an exception from a Granule Protection Check	When FEAT_RME is implemented

Value	Meaning	Link	Applies
`0b100000`	Instruction Abort from a lower Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from an Instruction Abort	
`0b100001`	Instruction Abort taken without a change in Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from an Instruction Abort	
`0b100010`	PC alignment fault exception.	ISS - an exception from an Illegal Execution state, or a PC or SP alignment fault	
`0b100100`	Data Abort from a lower Exception level, excluding Data Aborts taken to EL2 as a result of accesses generated associated with VNCR_EL2 as part of nested virtualization support. These Data Aborts might be generated from Exception levels in any Execution state. Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from a Data Abort	
`0b100101`	Data Abort without a change in Exception level, or Data Aborts taken to EL2 as a result of accesses generated associated with VNCR_EL2 as part of nested virtualization support. Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from a Data Abort	
`0b100110`	SP alignment fault exception.	ISS - an exception from an Illegal Execution state, or a PC or SP alignment fault	
`0b101000`	Trapped floating-point exception taken from AArch32 state. This EC value is valid if the implementation supports trapping of floating-point exceptions, otherwise it is reserved. Whether a floating-point implementation supports trapping of floating-point exceptions is IMPLEMENTATION DEFINED.	ISS - an exception from a trapped floating-point exception	When AArch32 is supported

Value	Meaning	Link	Applies
`0b101100`	Trapped floating-point exception taken from AArch64 state. This EC value is valid if the implementation supports trapping of floating-point exceptions, otherwise it is reserved. Whether a floating-point implementation supports trapping of floating-point exceptions is IMPLEMENTATION DEFINED.	ISS - an exception from a trapped floating-point exception	When AArch64 is supported
`0b101111`	SError interrupt.	ISS - an SError interrupt	
`0b110000`	Breakpoint exception from a lower Exception level.	ISS - an exception from a Breakpoint or Vector Catch debug exception	
`0b110001`	Breakpoint exception taken without a change in Exception level.	ISS - an exception from a Breakpoint or Vector Catch debug exception	
`0b110010`	Software Step exception from a lower Exception level.	ISS - an exception from a Software Step exception	
`0b110011`	Software Step exception taken without a change in Exception level.	ISS - an exception from a Software Step exception	
`0b110100`	Watchpoint from a lower Exception level, excluding Watchpoint Exceptions taken to EL2 as a result of accesses generated associated with VNCR_EL2 as part of nested virtualization support. These Watchpoint Exceptions might be generated from Exception levels using any Execution state.	ISS - an exception from a Watchpoint exception	
`0b110101`	Watchpoint exceptions without a change in Exception level, or Watchpoint exceptions taken to EL2 as a result of accesses generated associated with VNCR_EL2 as part of nested virtualization support.	ISS - an exception from a Watchpoint exception	
`0b111000`	BKPT instruction execution in AArch32 state.	ISS - an exception from execution of a Breakpoint instruction	When AArch32 is supported
`0b111010`	Vector Catch exception from AArch32 state. The only case where a Vector Catch exception is taken to an Exception level that is using AArch64 is when the exception is routed to EL2 and EL2 is using AArch64.	ISS - an exception from a Breakpoint or Vector Catch debug exception	When AArch32 is supported
`0b111100`	BRK instruction execution in AArch64 state.	ISS - an exception from execution of a Breakpoint instruction	When AArch64 is supported

All other EC values are reserved by Arm, and:

- Unused values in the range 0b000000 - 0b101100 (0x00 - 0x2C) are reserved for future use for synchronous exceptions.
- Unused values in the range 0b101101 - 0b111111 (0x2D - 0x3F) are reserved for future use, and might be used for synchronous or asynchronous exceptions.

The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [25]

Instruction Length for synchronous exceptions. Possible values of this bit are:

Value	Meaning
`0b0`	16-bit instruction trapped.
`0b1`	32-bit instruction trapped. This value is also used when the exception is one of the following: • An SError interrupt. • An Instruction Abort exception. • A PC alignment fault exception. • An SP alignment fault exception. • A Data Abort exception for which the value of the ISV bit is 0. • An Illegal Execution state exception. • Any debug exception except for Breakpoint instruction exceptions. For Breakpoint instruction exceptions, this bit has its standard meaning: – 0b0: 16-bit T32 BKPT instruction. – 0b1: 32-bit A32 BKPT instruction or A64 BRK instruction. • An exception reported using EC value 0b000000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS, bits [24:0]

Instruction Specific Syndrome. Architecturally, this field can be defined independently for each defined Exception class. However, in practice, some ISS encodings are used for more than one Exception class.

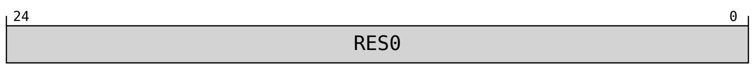
Typically, an ISS encoding has a number of subfields. When an ISS subfield holds a register number, the value returned in that field is the AArch64 view of the register number.

For an exception taken from AArch32 state, see 'Mapping of the general-purpose registers between the Execution states'.

If the AArch32 register descriptor is 0b1111, then:

- If the instruction that generated the exception was not UNPREDICTABLE, the field takes the value 0b11111.
- If the instruction that generated the exception was UNPREDICTABLE, the field takes an UNKNOWN value that must be either:
  - The AArch64 view of the register number of a register that might have been used at the Exception level from which the exception was taken.
  - The value 0b11111.

ISS encoding for exceptions with an unknown reason



Bits [24:0]

Reserved, RES0.

Additional information for exceptions with an unknown reason

When an exception is reported using this EC code the IL field is set to 1.

This EC code is used for all exceptions that are not covered by any other EC value. This includes exceptions that are generated in the following situations:

- The attempted execution of an instruction bit pattern that has no allocated instruction or that is not accessible at the current Exception level and Security state, including:
  - A read access using a System register pattern that is not allocated for reads or that does not permit reads at the current Exception level and Security state.
  - A write access using a System register pattern that is not allocated for writes or that does not permit writes at the current Exception level and Security state.
  - Instruction encodings that are unallocated.
  - Instruction encodings for instructions or System registers that are not implemented in the implementation.
- In Debug state, the attempted execution of an instruction bit pattern that is not accessible in Debug state.
- In Non-debug state, the attempted execution of an instruction bit pattern that is not accessible in Non-debug state.
- In AArch32 state, attempted execution of a short vector floating-point instruction.
- In an implementation that does not include Advanced SIMD and floating-point functionality, an attempted access to Advanced SIMD or floating-point functionality under conditions where that access would be permitted if that functionality was present. This includes the attempted execution of an Advanced SIMD or floating-point instruction, and attempted accesses to Advanced SIMD and floating-point System registers.
- An exception generated because of the value of one of the SCTLR_EL1.{ITD, SED, CP15BEN} control bits.
- Attempted execution of:
  - An HVC instruction when disabled by HCR_EL2.HCD or SCR_EL3.HCE.
  - An SMC instruction when disabled by SCR_EL3.SMD.
  - An HLT instruction when disabled by EDSCR.HDE.
- Attempted execution of an MSR or MRS instruction to access SP_EL0 when the value of SPSel.SP is 0.
- Attempted execution of an MSR or MRS instruction using a _EL12 register name when HCR_EL2.E2H == 0.
- Attempted execution, in Debug state, of:
  - A DCPS1 instruction when the value of HCR_EL2.TGE is 1 and EL2 is disabled or not implemented in the current Security state.
  - A DCPS2 instruction from EL1 or EL0 when EL2 is disabled or not implemented in the current Security state.
  - A DCPS3 instruction when the value of EDSCR.SDD is 1, or when EL3 is not implemented.
- When EL3 is using AArch64, attempted execution from Secure EL1 of an SRS instruction using R13_mon. See 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32'.
- In Debug state when the value of EDSCR.SDD is 1, the attempted execution at EL2, EL1, or EL0 of an instruction that is configured to trap to EL3.
- In AArch32 state, the attempted execution of an MRS (banked register) or an MSR (banked register) instruction to SPSR_mon, SP_mon, or LR_mon.
- An exception that is taken to EL2 because the value of HCR_EL2.TGE is 1 that, if the value of HCR_EL2.TGE was 0 would have been reported with an ESR_ELx.EC value of 0b000111.
- In Non-transactional state, attempted execution of a TCOMMIT instruction.

ISS encoding for an exception from a WF instruction*

24	23 20	19 10	9 5	4 3	2	1 0
CV	COND	RES0	RN	RES0	RV	TI

CV, bit [24]

Condition code valid.

Value	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:10]

Reserved, RES0.

RN, bits [9:5]

When FEAT_WFxT is implemented:

Register Number. Indicates the register number supplied for a WFET or WFIT instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [4:3]

Reserved, RES0.

RV, bit [2]

When FEAT_WFxT is implemented:

Register field Valid.

If TI[1] == 1, then this field indicates whether RN holds a valid register number for the register argument to the trapped WFET or WFIT instruction.

Value	Meaning
`0b0`	Register field invalid.
`0b1`	Register field valid.

If TI[1] == 0, then this field is RES0.

This field is set to 1 on a trap on WFET or WFIT.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TI, bits [1:0]

Trapped instruction. Possible values of this bit are:

Value	Meaning	Applies
`0b00`	WFI trapped.	
`0b01`	WFE trapped.	
`0b10`	WFIT trapped.	When FEAT_WFxT is implemented
`0b11`	WFET trapped.	When FEAT_WFxT is implemented

When FEAT_WFxT is implemented, this is a two bit field as shown. Otherwise, bit[1] is RES0.

The reset behavior of this field is:

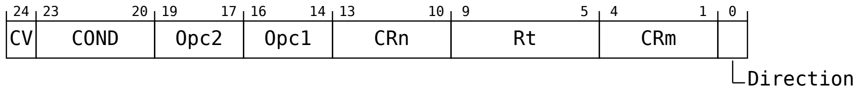
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a WF* instruction

The following fields describe configuration settings for generating this exception:

- SCTLR_EL1.{nTWE, nTWI}.
- HCR_EL2.{TWE, TWI}.
- SCR_EL3.{TWE, TWI}.

ISS encoding for an exception from an MCR or MRC access



CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc2, bits [19:17]

The Opc2 value from the issued instruction.

For a trapped VMRS access, holds the value 0b000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [16:14]

The Opc1 value from the issued instruction.

For a trapped VMRS access, holds the value 0b111.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

For a trapped VMRS access, holds the reg field from the VMRS instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

For a trapped VMRS access, holds the value 0b0000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
`0b0`	Write to System register space. MCR instruction.
`0b1`	Read from System register space. MRC or VMRS instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an MCR or MRC access

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000011:

- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- PMUSERENR_EL0.{ER, CR, SW, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- AMUSERENR_EL0.EN, for accesses to Activity Monitors registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TTLB, for execution of TLB maintenance instructions at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.{TSW, TPC, TPU} for execution of cache maintenance instructions at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TACR, for accesses to the Auxiliary Control Register at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TIDCP, for accesses to lockdown, DMA, and TCM operations at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.{TID1, TID2, TID3}, for accesses to ID registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- CPTR_EL2.TCPAC, for accesses to CPACR_EL1 or CPACR using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HSTR_EL2.T<n>, for accesses to System registers using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- CNTHCTL_EL2.EL1PCEN, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- CPTR_EL3.TCPAC, for accesses to CPACR from EL1 and EL2, and accesses to HCPTR from EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- For information on other traps using EC value 0b000011, see 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32'.
- If FEAT_FGT is implemented, MCR or MRC access to some registers at EL0, trapped to EL2.

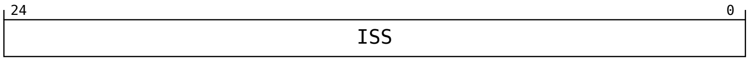
The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000101:

- CPACR_EL1.TTA for accesses to trace registers, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- MDSCR_EL1.TDCC, for accesses to the Debug Communications Channel (DCC) registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- If FEAT_FGT is implemented, MDCR_EL2.TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- HCR_EL2.TID0, for accesses to the JIDR register in the ID group 0 at EL0 and EL1 using AArch32, MRC access (coproc == 0b1110) trapped to EL2.
- CPTR_EL2.TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers DBGDRAR and DBGDSAR using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL2.TDOSA, for accesses to powerdown debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL2.TDA, for accesses to other debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- CPTR_EL3.TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.
- MDCR_EL3.TDOSA, for accesses to powerdown debug registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.
- MDCR_EL3.TDA, for accesses to other debug registers, using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001000:

- HCR_EL2.TID0, for accesses to the FPSID register in ID group 0 at EL1 using AArch32 state, VMRS access trapped to EL2.
- HCR_EL2.TID3, for accesses to registers in ID group 3 including MVFR0, MVFR1 and MVFR2, VMRS access trapped to EL2.

ISS encoding for an exception from an LD64B or ST64B instruction*

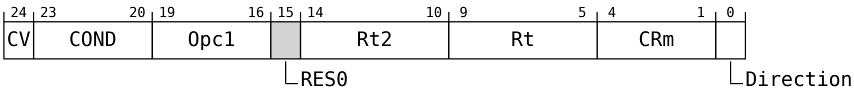


ISS, bits [24:0]

Value	Meaning
0b0000000000000000000000000	ST64BV instruction trapped.
0b0000000000000000000000001	ST64BV0 instruction trapped.
0b0000000000000000000000010	LD64B or ST64B instruction trapped.

All other values are reserved.

ISS encoding for an exception from an MCRR or MRRC access



CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [19:16]

The Opc1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [15]

Reserved, RES0.

Rt2, bits [14:10]

The Rt2 value from the issued instruction, the second general-purpose register used for the transfer.

If the Rt2 value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt2 value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the first general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
0b0	Write to System register space. MCRR instruction.
0b1	Read from System register space. MRRC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an MCRR or MRRC access

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000100:

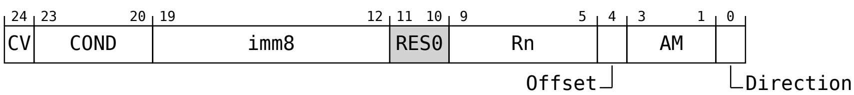
- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- PMUSERENR_EL0.{CR, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- AMUSERENR_EL0.{EN}, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- HSTR_EL2.T<n>, for accesses to System registers using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- CNTHCTL_EL2.{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- If FEAT_FGT is implemented, HDFGRTR_EL2.PMCCNTR_EL0 for MRRC access and HDFGWTR_EL2.PMCCNTR_EL0 for MCRR access to PMCCNTR at EL0, trapped to EL2.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001100:

- MDSCR_EL1.TDCC, for accesses to the Debug ROM registers DBGDSAR and DBGDRAR at EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers DBGDRAR and DBGDSAR using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL3.TDA, for accesses to debug registers, using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.
- CPACR_EL1.TTA for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- CPTR_EL2.TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- CPTR_EL3.TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.

If the Armv8-A architecture is implemented with an ETMv4 implementation, MCRR and MRRC accesses to trace registers are UNDEFINED and the resulting exception is higher priority than an exception due to these traps.

ISS encoding for an exception from an LDC or STC instruction



CV, bit [24]

Condition code valid.

Value	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

imm8, bits [19:12]

The immediate value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:10]

Reserved, RES0.

Rn, bits [9:5]

The Rn value from the issued instruction, the general-purpose register used for the transfer.

If the Rn value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rn value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

This field is valid only when AM[2] is 0, indicating an immediate form of the LDC or STC instruction. When AM[2] is 1, indicating a literal form of the LDC or STC instruction, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Offset, bit [4]

Indicates whether the offset is added or subtracted:

Value	Meaning
0b0	Subtract offset.
0b1	Add offset.

This bit corresponds to the U bit in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

AM, bits [3:1]

Addressing mode. The permitted values of this field are:

Value	Meaning
0b000	Immediate unindexed.
0b001	Immediate post-indexed.
0b010	Immediate offset.
0b011	Immediate pre-indexed.
0b100	For a trapped STC instruction or a trapped T32 LDC instruction this encoding is reserved.
0b110	For a trapped STC instruction, this encoding is reserved.

The values 0b101 and 0b111 are reserved. The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in System and memory-mapped registers and translation table entries'.

Bit [2] in this subfield indicates the instruction form, immediate or literal.

Bits [1:0] in this subfield correspond to the bits {P, W} in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
`0b0`	Write to memory. STC instruction.
`0b1`	Read from memory. LDC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an LDC or STC instruction

The following fields describe the configuration settings for the traps that are reported using EC value 0b000110:

- MDSCR_EL1.TDCC, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint trapped to EL1 or EL2.
- MDCR_EL2.TDA, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint MCR or MRC access trapped to EL2.
- MDCR_EL3.TDA, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint MCR or MRC access trapped to EL3.
- If FEAT_FGT is implemented, MDCR_EL2.TDCC for LDC and STC accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.

ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps

24	23–20	19–0
CV	COND	RES0

The accesses covered by this trap include:

- Execution of SVE or Advanced SIMD and floating-point instructions.
- Accesses to the Advanced SIMD and floating-point System registers.
- Execution of SME instructions.

For an implementation that does not include either SVE or support for Advanced SIMD and floating-point, the exception is reported using the EC value 0b000000.

CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.

Value	Meaning
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:0]

Reserved, RES0.

Additional information for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps

The following fields describe the configuration settings for the traps that are reported using EC value 0b000111:

- CPACR_EL1.FPEN, for accesses to SIMD and floating-point registers trapped to EL1.
- CPTR_EL2.FPEN and CPTR_EL2.TFP, for accesses to SIMD and floating-point registers trapped to EL2.
- CPTR_EL3.TFP, for accesses to SIMD and floating-point registers trapped to EL3.

ISS encoding for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ

24–0
RES0

The accesses covered by this trap include:

- Execution of SVE instructions when the PE is not in Streaming SVE mode.
- Accesses to the SVE System registers, ZCR_ELx.

For an implementation that does not include SVE, the exception is reported using the EC value 0b000000.

Bits [24:0]

Reserved, RES0.

Additional information for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ

The following fields describe the configuration settings for the traps that are reported using EC value 0b011001:

- CPACR_EL1.ZEN, for execution of SVE instructions and accesses to SVE registers at EL0 or EL1, trapped to EL1.
- CPTR_EL2.ZEN and CPTR_EL2.TZ, for execution of SVE instructions and accesses to SVE registers at EL0, EL1, or EL2, trapped to EL2.
- CPTR_EL3.EZ, for execution of SVE instructions and accesses to SVE registers from all Exception levels, trapped to EL3.

ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault

24–0
RES0

Bits [24:0]

Reserved, RES0.

Additional information for an exception from an Illegal Execution state, or a PC or SP alignment fault

There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about these exceptions, see ‘The Illegal Execution state exception’ and ‘PC alignment checking’.

‘SP alignment checking’ describes the configuration settings for generating SP alignment fault exceptions.

ISS encoding for an exception from HVC or SVC instruction execution

24–16	15–0
RES0	imm16

Bits [24:16]

Reserved, RES0.

imm16, bits [15:0]

The value of the immediate field from the HVC or SVC instruction.

For an HVC instruction, and for an A64 SVC instruction, this is the value of the imm16 field of the issued instruction.

For an A32 or T32 SVC instruction:

- If the instruction is unconditional, then:
  - For the T32 instruction, this field is zero-extended from the imm8 field of the instruction.
  - For the A32 instruction, this field is the bottom 16 bits of the imm24 field of the instruction.
- If the instruction is conditional, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from HVC or SVC instruction execution

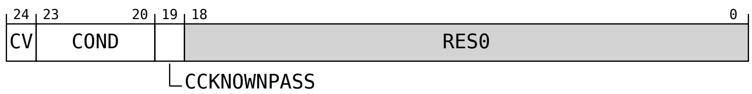
In AArch32 state, the HVC instruction is unconditional, and a conditional SVC instruction generates an exception only if it passes its condition code check. Therefore, the syndrome information for these exceptions does not require conditionality information.

For T32 and A32 instructions, see 'SVC' and 'HVC'.

For A64 instructions, see 'SVC' and 'HVC'.

If FEAT_FGT is implemented, HFGITR_EL2.{SVC_EL1, SVC_EL0} control fine-grained traps on SVC execution.

ISS encoding for an exception from SMC instruction execution in AArch32 state



For an SMC instruction that completes normally and generates an exception that is taken to EL3, the ISS encoding is RES0.

For an SMC instruction that is trapped to EL2 from EL1 because HCR_EL2.TSC is 1, the ISS encoding is as shown in the diagram.

CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

This field is valid only if CCKNOWNPASS is 1, otherwise it is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

This field is valid only if CCKNOWNPASS is 1, otherwise it is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CCKNOWNPASS, bit [19]

Indicates whether the instruction might have failed its condition code check.

Value	Meaning
`0b0`	The instruction was unconditional, or was conditional and passed its condition code check.
`0b1`	The instruction was conditional, and might have failed its condition code check.

In an implementation in which an SMC instruction that fails it code check is not trapped, this field can always return the value 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [18:0]

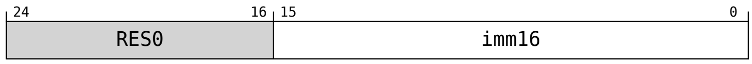
Reserved, RES0.

Additional information for an exception from SMC instruction execution in AArch32 state

HCR_EL2.TSC describes the configuration settings for trapping SMC instructions to EL2.

'System calls' describes the case where these exceptions are trapped to EL3.

ISS encoding for an exception from SMC instruction execution in AArch64 state



Bits [24:16]

Reserved, RES0.

imm16, bits [15:0]

The value of the immediate field from the issued SMC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from SMC instruction execution in AArch64 state

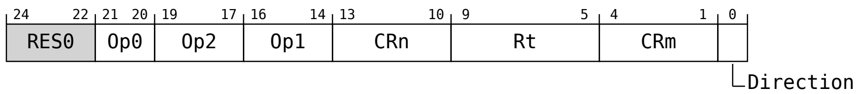
The value of ISS[24:0] described here is used both:

- When an SMC instruction is trapped from EL1 modes.
- When an SMC instruction is not trapped, so completes normally and generates an exception that is taken to EL3.

HCR_EL2.TSC describes the configuration settings for trapping SMC from EL1 modes.

'System calls' describes the case where these exceptions are trapped to EL3.

ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state



Bits [24:22]

Reserved, RES0.

Op0, bits [21:20]

The Op0 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op2, bits [19:17]

The Op2 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op1, bits [16:14]

The Op1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
0b0	Write access, including MSR instructions.
0b1	Read access, including MRS instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from MSR, MRS, or System instruction execution in AArch64 state

For exceptions caused by System instructions, see ‘System instructions’ subsection of ‘Branches, exception generating and System instructions’ for the encoding values returned by an instruction.

The following fields describe configuration settings for generating the exception that is reported using EC value 0b011000:

- SCTLR_EL1.UCI, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- SCTLR_EL1.UCT, for accesses to CTR_EL0 using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- SCTLR_EL1.DZE, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- SCTLR_EL1.UMA, for accesses to the PSTATE interrupt masks using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- CPACR_EL1.TTA, for accesses to the trace registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- MDSCR_EL1.TDCC, for accesses to the Debug Communications Channel (DCC) registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- If FEAT_FGT is implemented, MDCR_EL2.TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN} accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.

- PMUSERENR_EL0.{ER, CR, SW, EN}, for accesses to the Performance Monitor registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- AMUSERENR_EL0.EN, for accesses to Activity Monitors registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TDZ, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TTLB, for execution of TLB maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{TSW, TPC, TPU}, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TACR, for accesses to the Auxiliary Control Register, ACTLR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TIDCP, for accesses to lockdown, DMA, and TCM operations using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{TID1, TID2, TID3}, for accesses to ID group 1, ID group 2 or ID group 3 registers, using AArch64 state, MSR or MRS access trapped to EL2.
- CPTR_EL2.TCPAC, for accesses to CPACR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.
- CPTR_EL2.TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TTRF, for accesses to the trace filter control register, TRFCR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDOSA, for accesses to powerdown debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- CNTHCTL_EL2.{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDA, for accesses to debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.APK, for accesses to Pointer authentication key registers. using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{NV, NV1}, for Nested virtualization register access, using AArch64 state, MSR or MRS access, trapped to EL2.
- HCR_EL2.AT, for execution of AT S1E* instructions, using AArch64 state, MSR or MRS access, trapped to EL2.
- HCR_EL2.{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access, trapped to EL2.
- SCR_EL3.APK, for accesses to Pointer authentication key registers, using AArch64 state, MSR or MRS access trapped to EL3.
- SCR_EL3.ST, for accesses to the Counter-timer Physical Secure timer registers, using AArch64 state, MSR or MRS access trapped to EL3.
- SCR_EL3.{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TCPAC, for accesses to CPTR_EL2 and CPACR_EL1 using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TTRF, for accesses to the trace filter control registers, TRFCR_EL1 and TRFCR_EL2, using AArch64 state, MSR or MRS access trapped to EL3.

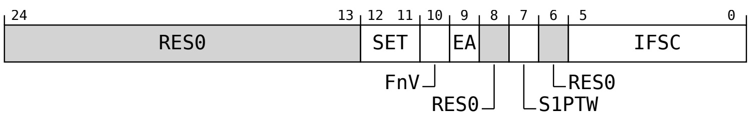
- MDCR_EL3.TDA, for accesses to debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TDOSA, for accesses to powerdown debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access, trapped to EL3.
- If FEAT_EVT is implemented, the following registers control traps for EL1 and EL0 Cache controls that use this EC value:
  - HCR_EL2.{TTLBOS, TTLBIS, TICAB, TOCU, TID4}.
  - HCR2.{TTLBIS, TICAB, TOCU, TID4}.
- If FEAT_FGT is implemented:
  - SCR_EL3.FGTEn, for accesses to the fine-grained trap registers, MSR or MRS access at EL2 trapped to EL3.
  - HFGRTR_EL2 for reads and HFGWTR_EL2 for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 trapped to EL2.
  - HFGITR_EL2 for execution of system instructions, MSR or MRS access trapped to EL2
  - HDFGRTR_EL2 for reads and HDFGWTR_EL2 for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 state trapped to EL2.
  - HAFGRTR_EL2 for reads of Activity Monitor counters, using AArch64 state, MRS access at EL0 and EL1 trapped to EL2.
- If FEAT_RNG_TRAP is implemented:
  - SCR_EL3.TRNDR for reads of RNDR and RNDRRS using AArch64 state, MRS access trapped to EL3.
- If FEAT_SME is implemented:
  - CPTR_EL3.ESM, for MSR or MRS accesses to SMPRI_EL1 at EL1, EL2, and EL3, trapped to EL3.
  - CPTR_EL3.ESM, for MSR or MRS accesses to SMPRIMAP_EL2 at EL2 and EL3, trapped to EL3.
  - SCTLR_EL1.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, trapped to EL1 or EL2.
  - SCTLR_EL2.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, trapped to EL2.
  - SCR_EL3.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, EL1, and EL2, trapped to EL3.

ISS encoding for an IMPLEMENTATION DEFINED exception to EL3

IMPLEMENTATION DEFINED, bits [24:0]

IMPLEMENTATION DEFINED

ISS encoding for an exception from an Instruction Abort



Bits [24:13]

Reserved, RES0.

SET, bits [12:11]

When FEAT_RAS is implemented:

Synchronous Error Type. When IFSC is 0b010000, describes the PE error state after taking the Instruction Abort exception.

Value	Meaning
`0b00`	Recoverable state (UER).
`0b10`	Uncontainable (UC).
`0b11`	Restartable state (UEO).

All other values are reserved.

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

Value	Meaning
`0b0`	FAR is valid.
`0b1`	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [8]

Reserved, RES0.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

Value	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

Value	Meaning	Applies
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	

Value	Meaning	Applies
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b110000	TLB conflict abort.	

Value	Meaning	Applies
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

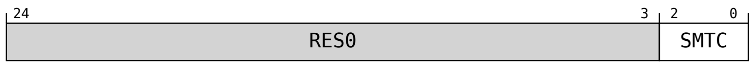
Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception due to SME functionality



The accesses covered by this trap include:

- Execution of SME instructions.
- Execution of SVE and Advanced SIMD instructions, when the PE is in Streaming SVE mode.
- Direct accesses of SVCR, SMCR_EL1, SMCR_EL2, SMCR_EL3.

Bits [24:3]

Reserved, RES0.

SMTC, bits [2:0]

SME Trap Code. Identifies the reason for instruction trapping.

Value	Meaning
0b000	Access to SME functionality trapped as a result of CPACR_EL1.SMEN, CPTR_EL2.SMEN, CPTR_EL2.TSM, or CPTR_EL3.ESM, that is not reported using EC 0b000000.
0b001	Advanced SIMD, SVE, or SVE2 instruction trapped because PSTATE.SM is 1.
0b010	SME instruction trapped because PSTATE.SM is 0.
0b011	SME instruction trapped because PSTATE.ZA is 0.

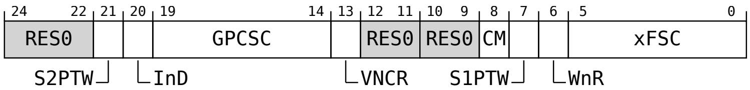
All other values are reserved.

Additional information for an exception due to SME functionality

The following fields describe the configuration settings for the traps that are reported using the EC value 0b011101:

- CPACR_EL1.SMEN, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR and SMCR_EL1 System registers at EL1 and EL0, trapped to EL1 or EL2.
- CPTR_EL2.SMEN and CPTR_EL2.TSM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR, SMCR_EL1, SMCR_EL2 at EL2, EL1, or EL0, trapped to EL2.
- CPTR_EL3.ESM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR, SMCR_EL1, SMCR_EL2, SMCR_EL3 from all Exception levels and any Security state, trapped to EL3.

ISS encoding for an exception from a Granule Protection Check



Bits [24:22]

Reserved, RES0.

S2PTW, bit [21]

Indicates whether the Granule Protection Check exception was on an access made for a stage 2 translation table walk.

Value	Meaning
`0b0`	Fault not on a stage 2 translation table walk.
`0b1`	Fault on a stage 2 translation table walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

InD, bit [20]

Indicates whether the Granule Protection Check exception was on an instruction or data access.

Value	Meaning
`0b0`	Data access.
`0b1`	Instruction access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GPCSC, bits [19:14]

Granule Protection Check Status Code.

Value	Meaning
`0b000000`	GPT address size fault at level 0.

Value	Meaning
`0b000001`	GPT address size fault at level 1.
`0b000100`	GPT walk fault at level 0.
`0b000101`	GPT walk fault at level 1.
`0b001100`	Granule protection fault at level 0.
`0b001101`	Granule protection fault at level 1.
`0b010100`	Synchronous External abort on GPT fetch at level 0.
`0b010101`	Synchronous External abort on GPT fetch at level 1.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [13]

When FEAT_NV2 is implemented

VNCR, bit [13]

Indicates that the fault came from use of VNCR_EL2 register by EL1 code.

Value	Meaning
`0b0`	The fault was not generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.
`0b1`	The fault was generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

When InD is '1', this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

Bit [13]

Reserved, RES0.

Bits [12:11]

Reserved, RES0.

Bits [10:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

Value	Meaning
`0b0`	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
`0b1`	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

Indicates whether the Granule Protection Check exception was on an access for stage 2 translation for a stage 1 translation table walk:

Value	Meaning
`0b0`	Fault not on a stage 2 translation for a stage 1 translation table walk.
`0b1`	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

Value	Meaning
`0b0`	Abort caused by an instruction reading from a memory location.
`0b1`	Abort caused by an instruction writing to a memory location.

When InD is ‘1’, this field is RES0.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

xFSC, bits [5:0]

Instruction or Data Fault Status Code.

Value	Meaning	Applies
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

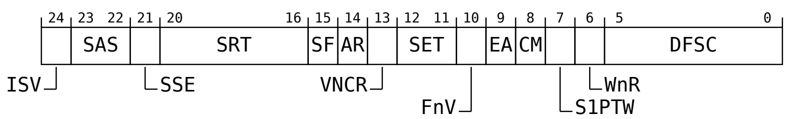
Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a Data Abort



When FEAT_LS64 is implemented, if a memory access generated by an ST64BV or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this ISS encoding includes ISS2, bits[36:32].

ISV, bit [24]

Instruction Syndrome Valid. Indicates whether the syndrome information in ISS[23:14] is valid.

Value	Meaning
0b0	No valid instruction syndrome. ISS[23:14] are RES0.
0b1	ISS[23:14] hold a valid instruction syndrome.

In ESR_EL2, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For other faults reported in ESR_EL2, ISV is 0 except for the following stage 2 aborts:

- AArch64 loads and stores of a single general-purpose register (including the register specified with 0b11111, including those with Acquire/Release semantics, but excluding Load Exclusive or Store Exclusive and excluding those with writeback).
- AArch32 instructions where the instruction:
  - Is an LDR, LDA, LDRT, LDRSH, LDRSHT, LDRH, LDAH, LDRHT, LDRSB, LDRSBT, LDRB, LDAB, LDRBT, STR, STL, STRT, STRH, STLH, STRHT, STRB, STLB, or STRBT instruction.
  - Is not performing register writeback.
  - Is not using R15 as a source or destination register.

For these stage 2 aborts, ISV is UNKNOWN if the exception was generated in Debug state in memory access mode, and otherwise indicates whether ISS[23:14] hold a valid syndrome.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault. ISV is 0 for all other faults reported in ESR_EL1 or ESR_EL3.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

For ISS reporting, a stage 2 abort on a stage 1 translation table walk does not return a valid instruction syndrome, and therefore ISV is 0 for these aborts.

When FEAT_RAS is not implemented, it is IMPLEMENTATION DEFINED whether ISV is set to 1 or 0 on a synchronous External abort on a stage 2 translation table walk.

When FEAT_MTE2 is implemented, for a synchronous Tag Check Fault abort taken to ELx, ESR_ELx.FNV is 0 and FAR_ELx is valid.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SAS, bits [23:22]

When ISV == 1:

Syndrome Access Size. Indicates the size of the access attempted by the faulting operation.

Value	Meaning
0b00	Byte
0b01	Halfword
0b10	Word
0b11	Doubleword

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SSE, bit [21]

When ISV == 1:

Syndrome Sign Extend. For a byte, halfword, or word load operation, indicates whether the data item must be sign extended.

Value	Meaning
`0b0`	Sign-extension not required.
`0b1`	Data item must be sign-extended.

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

For all other operations, this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SRT, bits [20:16]

When ISV == 1:

Syndrome Register Transfer. The register number of the Wt/Xt/Rt operand of the faulting instruction.

If the exception was taken from an Exception level that is using AArch32, then this is the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states'.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SF, bit [15]

When ISV == 1:

Width of the register accessed by the instruction is Sixty-Four.

Value	Meaning
`0b0`	Instruction loads/stores a 32-bit wide register.
`0b1`	Instruction loads/stores a 64-bit wide register.

This field specifies the register width identified by the instruction, not the Execution state.

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

AR, bit [14]

When ISV == 1:

Acquire/Release.

Value	Meaning
`0b0`	Instruction did not have acquire/release semantics.
`0b1`	Instruction did have acquire/release semantics.

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bit [13]

When FEAT_NV2 is implemented

VNCR, bit [13]

Indicates that the fault came from use of VNCR_EL2 register by EL1 code.

Value	Meaning
`0b0`	The fault was not generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.
`0b1`	The fault was generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

Bit [13]

Reserved, RES0.

Bits [12:11]

When FEAT_RAS is implemented and FEAT_LS64 is not implemented

SET, bits [12:11]

Synchronous Error Type. When DFSC is 0b010000, describes the PE error state after taking the Data Abort exception.

Value	Meaning
`0b00`	Recoverable state (UER).
`0b10`	Uncontainable (UC).
`0b11`	Restartable state (UEO).

All other values are reserved.

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

This field is valid only if the DFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_LS64 is implemented

LST, bits [12:11]

Load/Store Type. Used when an LD64B, ST64B, ST64BV, or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

Value	Meaning
`0b01`	An ST64BV instruction generated the Data Abort.
`0b10`	An LD64B or ST64B instruction generated the Data Abort.
`0b11`	An ST64BV0 instruction generated the Data Abort.

All other values are reserved.

This field is valid only if the DFSC code is 0b110101. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

Value	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the DFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

Value	Meaning
0b0	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

Value	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.

Value	Meaning
`0b1`	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

Value	Meaning
`0b0`	Abort caused by an instruction reading from a memory location.
`0b1`	Abort caused by an instruction writing to a memory location.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DFSC, bits [5:0]

Data Fault Status Code.

Value	Meaning	Applies
`0b000000`	Address size fault, level 0 of translation or translation table base register.	
`0b000001`	Address size fault, level 1.	
`0b000010`	Address size fault, level 2.	
`0b000011`	Address size fault, level 3.	
`0b000100`	Translation fault, level 0.	
`0b000101`	Translation fault, level 1.	
`0b000110`	Translation fault, level 2.	

Value	Meaning	Applies
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Synchronous Tag Check Fault.	When FEAT_MTE2 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented

Value	Meaning	Applies
`0b100001`	Alignment fault.	
`0b100011`	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
`0b100100`	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
`0b100101`	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
`0b100110`	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
`0b100111`	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
`0b101000`	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
`0b101001`	Address size fault, level -1.	When FEAT_LPA2 is implemented
`0b101011`	Translation fault, level -1.	When FEAT_LPA2 is implemented
`0b110000`	TLB conflict abort.	
`0b110001`	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
`0b110100`	IMPLEMENTATION DEFINED fault (Lockdown).	
`0b110101`	IMPLEMENTATION DEFINED fault (Unsupported Exclusive or Atomic access).	

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

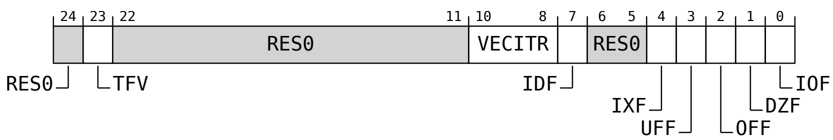
Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a trapped floating-point exception



Bit [24]

Reserved, RES0.

TFV, bit [23]

Trapped Fault Valid bit. Indicates whether the IDF, IXF, UFF, OFF, DZF, and IOF bits hold valid information about trapped floating-point exceptions.

Value	Meaning
`0b0`	The IDF, IXF, UFF, OFF, DZF, and IOF bits do not hold valid information about trapped floating-point exceptions and are UNKNOWN.
`0b1`	One or more floating-point exceptions occurred during an operation performed while executing the reported instruction. The IDF, IXF, UFF, OFF, DZF, and IOF bits indicate trapped floating-point exceptions that occurred. For more information, see 'Floating-point exceptions and exception traps'.

It is IMPLEMENTATION DEFINED whether this field is set to 0 on an exception generated by a trapped floating-point exception from an instruction that is performing floating-point operations on more than one lane of a vector.

This is not a requirement. Implementations can set this field to 1 on a trapped floating-point exception from an instruction and return valid information in the {IDF, IXF, UFF, OFF, DZF, IOF} fields.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [22:11]

Reserved, RES0.

VECITR, bits [10:8]

For a trapped floating-point exception from an instruction executed in AArch32 state this field is RES1.

For a trapped floating-point exception from an instruction executed in AArch64 state this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IDF, bit [7]

Input Denormal floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Input denormal floating-point exception has not occurred.
`0b1`	Input denormal floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

IXF, bit [4]

Inexact floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Inexact floating-point exception has not occurred.
`0b1`	Inexact floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

UFF, bit [3]

Underflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Underflow floating-point exception has not occurred.
`0b1`	Underflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

OFF, bit [2]

Overflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Overflow floating-point exception has not occurred.
`0b1`	Overflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DZF, bit [1]

Divide by Zero floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Divide by Zero floating-point exception has not occurred.
`0b1`	Divide by Zero floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IOF, bit [0]

Invalid Operation floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Invalid Operation floating-point exception has not occurred.
`0b1`	Invalid Operation floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

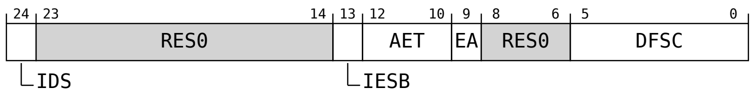
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a trapped floating-point exception

In an implementation that supports the trapping of floating-point exceptions:

- From an Exception level using AArch64, the FPCR.{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.
- From an Exception level using AArch32, the FPSCR.{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.

ISS encoding for an SError interrupt



IDS, bit [24]

IMPLEMENTATION DEFINED syndrome.

Value	Meaning
`0b0`	Bits [23:0] of the ISS field holds the fields described in this encoding. If FEAT_RAS is not implemented, bits [23:0] of the ISS field are RES0.
`0b1`	Bits [23:0] of the ISS field holds IMPLEMENTATION DEFINED syndrome information that can be used to provide additional information about the SError interrupt.

This field was previously called ISV.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [23:14]

Reserved, RES0.

IESB, bit [13]

When FEAT_IESB is implemented:

Implicit error synchronization event.

Value	Meaning
0b0	The SError interrupt was either not synchronized by the implicit error synchronization event or not taken immediately.
0b1	The SError interrupt was synchronized by the implicit error synchronization event and taken immediately.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

AET, bits [12:10]

When FEAT_RAS is implemented:

Asynchronous Error Type.

When DFSC is 0b010001, describes the PE error state after taking the SError interrupt exception.

Value	Meaning
0b000	Uncontainable (UC).
0b001	Unrecoverable state (UEU).
0b010	Restartable state (UEO).
0b011	Recoverable state (UER).
0b110	Corrected (CE).

All other values are reserved.

If multiple errors are taken as a single SError interrupt exception, the overall PE error state is reported.

Software can use this information to determine what recovery might be possible. The recovery software must also examine any implemented fault records to determine the location and extent of the error.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EA, bit [9]

When FEAT_RAS is implemented:

External abort type. When DFSC is 0b010001, provides an IMPLEMENTATION DEFINED classification of External aborts.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [8:6]

Reserved, RES0.

DFSC, bits [5:0]

When FEAT_RAS is implemented:

Data Fault Status Code.

Value	Meaning
0b000000	Uncategorized error.
0b010001	Asynchronous SError interrupt.

All other values are reserved.

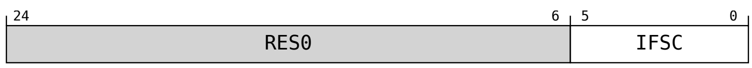
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

ISS encoding for an exception from a Breakpoint or Vector Catch debug exception



Bits [24:6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

Value	Meaning
0b100010	Debug exception.

The reset behavior of this field is:

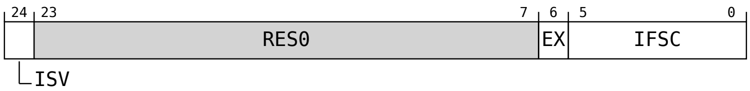
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a Breakpoint or Vector Catch debug exception

For more information about generating these exceptions:

- For exceptions from AArch64, see ‘Breakpoint exceptions’.
- For exceptions from AArch32, see ‘Breakpoint exceptions’ and ‘Vector Catch exceptions’.

ISS encoding for an exception from a Software Step exception



ISV, bit [24]

Instruction syndrome valid. Indicates whether the EX bit, ISS[6], is valid, as follows:

Value	Meaning
0b0	EX bit is RES0.
0b1	EX bit is valid.

See the EX bit description for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [23:7]

Reserved, RES0.

EX, bit [6]

Exclusive operation. If the ISV bit is set to 1, this bit indicates whether a Load-Exclusive instruction was stepped.

Value	Meaning
0b0	An instruction other than a Load-Exclusive instruction was stepped.
0b1	A Load-Exclusive instruction was stepped.

If the ISV bit is set to 0, this bit is RES0, indicating no syndrome data is available.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IFSC, bits [5:0]

Instruction Fault Status Code.

Value	Meaning
0b100010	Debug exception.

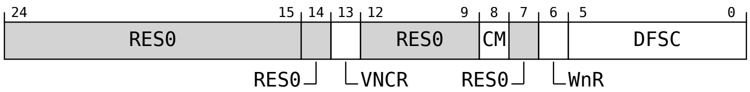
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a Software Step exception

For more information about generating these exceptions, see 'Software Step exceptions'.

ISS encoding for an exception from a Watchpoint exception



Bits [24:15]

Reserved, RES0.

Bit [14]

Reserved, RES0.

Bit [13]

When FEAT_NV2 is implemented

VNCR, bit [13]

Indicates that the watchpoint came from use of VNCR_EL2 register by EL1 code.

Value	Meaning
0b0	The watchpoint was not generated by the use of VNCR_EL2 by EL1 code.
0b1	The watchpoint was generated by the use of VNCR_EL2 by EL1 code.

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

Bit [13]

Reserved, RES0.

Bits [12:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Watchpoint exception came from a cache maintenance or address translation instruction:

Value	Meaning
`0b0`	The Watchpoint exception was not generated by the execution of one of the System instructions identified in the description of value 1.
`0b1`	The Watchpoint exception was generated by either the execution of a cache maintenance instruction or by a synchronous Watchpoint exception on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as a cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [7]

Reserved, RES0.

WnR, bit [6]

Write not Read. Indicates whether the Watchpoint exception was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

Value	Meaning
`0b0`	Watchpoint exception caused by an instruction reading from a memory location.
`0b1`	Watchpoint exception caused by an instruction writing to a memory location.

For Watchpoint exceptions on cache maintenance and address translation instructions, this bit always returns a value of 1.

For Watchpoint exceptions from an atomic instruction, this field is set to 0 if a read of the location would have generated the Watchpoint exception, otherwise it is set to 1.

If multiple watchpoints match on the same access, it is UNPREDICTABLE which watchpoint generates the Watchpoint exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DFSC, bits [5:0]

Data Fault Status Code.

Value	Meaning
`0b100010`	Debug exception.

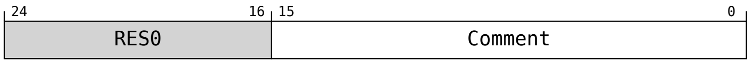
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a Watchpoint exception

For more information about generating these exceptions, see 'Watchpoint exceptions'.

ISS encoding for an exception from execution of a Breakpoint instruction



Bits [24:16]

Reserved, RES0.

Comment, bits [15:0]

Set to the instruction comment field value, zero extended as necessary.

For the AArch32 BKPT instructions, the comment field is described as the immediate field.

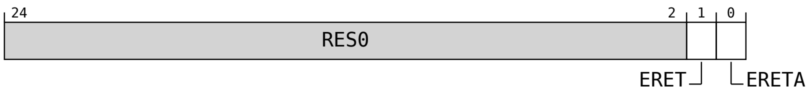
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from execution of a Breakpoint instruction

For more information about generating these exceptions, see 'Breakpoint instruction exceptions'.

ISS encoding for an exception from an ERET, ERETAA, or ERETAB instruction



This EC value applies when FEAT_FGT is implemented, or when HCR_EL2.NV is 1.

Bits [24:2]

Reserved, RES0.

ERET, bit [1]

Indicates whether an ERET or ERETA* instruction was trapped to EL2.

Value	Meaning
`0b0`	ERET instruction trapped to EL2.
`0b1`	ERETAA or ERETAB instruction trapped to EL2.

If this bit is 0, the ERETA field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ERETA, bit [0]

Indicates whether an ERETAA or ERETAB instruction was trapped to EL2.

Value	Meaning
0b0	ERETAA instruction trapped to EL2.
0b1	ERETAB instruction trapped to EL2.

When the ERET field is 0, this bit is RES0.

The reset behavior of this field is:

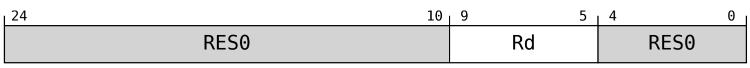
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an ERET, ERETAA, or ERETAB instruction

For more information about generating these exceptions, see HCR_EL2.NV.

If FEAT_FGT is implemented, HFGITR_EL2.ERET controls fine-grained trap exceptions from ERET, ERETAA and ERETAB execution.

ISS encoding for an exception from a TSTART instruction



Bits [24:10]

Reserved, RES0.

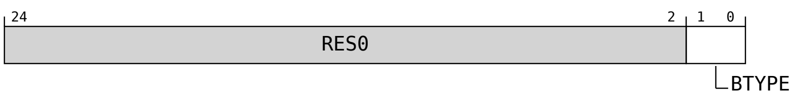
Rd, bits [9:5]

The Rd value from the issued instruction, the general purpose register used for the destination.

Bits [4:0]

Reserved, RES0.

ISS encoding for an exception from Branch Target Identification instruction



Bits [24:2]

Reserved, RES0.

BTYPE, bits [1:0]

This field is set to the PSTATE.BTYPE value that generated the Branch Target Exception.

Additional information for an exception from Branch Target Identification instruction

For more information about generating these exceptions, see 'The AArch64 application level programmers model'.

ISS encoding for an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 || SCR_EL3.API == 0

Bits [24:0]

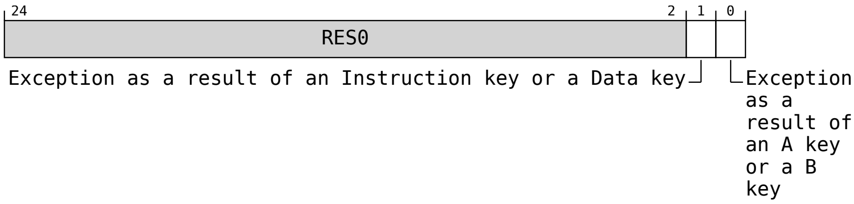
Reserved, RES0.

Additional information for an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 || SCR_EL3.API == 0

For more information about generating these exceptions, see:

- HCR_EL2.API, for exceptions from Pointer authentication instructions, using AArch64 state, trapped to EL2.
- SCR_EL3.API, for exceptions from Pointer authentication instructions, using AArch64 state, trapped to EL3.

ISS encoding for an exception from a Pointer Authentication instruction authentication failure



Bits [24:2]

Reserved, RES0.

Bit [1]

This field indicates whether the exception is as a result of an Instruction key or a Data key.

Value	Meaning
`0b0`	Instruction Key.
`0b1`	Data Key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [0]

This field indicates whether the exception is as a result of an A key or a B key.

Value	Meaning
`0b0`	A key.
`0b1`	B key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a Pointer Authentication instruction authentication failure

The following instructions generate an exception when the Pointer Authentication Code (PAC) is incorrect:

- AUTIASP, AUTIAZ, AUTIA1716.
- AUTIBSP, AUTIBZ, AUTIB1716.
- AUTIA, AUTDA, AUTIB, AUTDB.
- AUTIZA, AUTIZB, AUTDZA, AUTDZB.

It is IMPLEMENTATION DEFINED whether the following instructions generate an exception directly from the authorization failure, rather than changing the address in a way that will generate a Translation fault when the address is accessed:

- RETAA, RETAB.
- BRAA, BRAB, BLRAA, BLRAB.
- BRAAZ, BRABZ, BLRAAZ, BLRABZ.
- ERETAA, ERETAB.
- LDRAA, LDRAB, whether the authenticated address is written back to the base register or not.

Accessing the ESR_EL2

When HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic ESR_EL2 or ESR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, ESR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return ESR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return ESR_EL2;
elsif PSTATE.EL == EL3 then
    return ESR_EL2;
```

MSR ESR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
```

```
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        ESR_EL1 = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    ESR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    ESR_EL2 = X[t];
```

MRS <Xt>, ESR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGRTR_EL2.ESR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x138];
    else
        return ESR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return ESR_EL2;
    else
        return ESR_EL1;
elsif PSTATE.EL == EL3 then
    return ESR_EL1;
```

MSR ESR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGWTR_EL2.ESR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x138] = X[t];
    else
        ESR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        ESR_EL2 = X[t];
    else
        ESR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ESR_EL1 = X[t];
```

A2.1.7 ESR_EL3, Exception Syndrome Register (EL3)

The ESR_EL3 characteristics are:

Purpose

Holds syndrome information for an exception taken to EL3.

Attributes

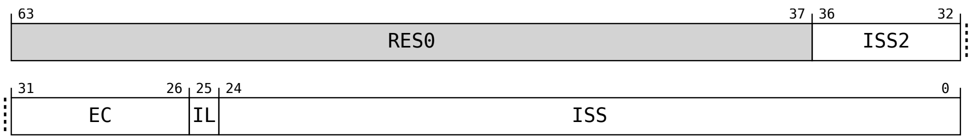
ESR_EL3 is a 64-bit register.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to ESR_EL3 are UNDEFINED.

Field descriptions

The ESR_EL3 bit assignments are:



ESR_EL3 is made UNKNOWN as a result of an exception return from EL3.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to EL3, the value of ESR_EL3 is UNKNOWN. The value written to ESR_EL3 must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not UNPREDICTABLE at that Exception level, in order to avoid the possibility of a privilege violation.

Bits [63:37]

Reserved, RES0.

ISS2, bits [36:32]

When FEAT_LS64 is implemented:

If a memory access generated by an ST64BV or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

For any other Data Abort, this field is RES0.

Otherwise:

RES0

EC, bits [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about.

For each EC value, the table references a subsection that gives information about:

- The cause of the exception, for example the configuration required to enable the trap.
- The encoding of the associated ISS.

Possible values of the EC field are:

Value	Meaning	Link	Applies
0b000000	Unknown reason.	ISS - exceptions with an unknown reason	
0b000001	Trapped WF* instruction execution. Conditional WF* instructions that fail their condition code check do not cause an exception.	ISS - an exception from a WF* instruction	
0b000011	Trapped MCR or MRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS - an exception from an MCR or MRC access	When AArch32 is supported
0b000100	Trapped MCRR or MRRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS - an exception from an MCRR or MRRC access	When AArch32 is supported
0b000101	Trapped MCR or MRC access with (coproc==0b1110).	ISS - an exception from an MCR or MRC access	When AArch32 is supported
0b000110	Trapped LDC or STC access. The only architected uses of these instruction are: • An STC to write data to memory from DBGDTRRXint. • An LDC to read data from memory to DBGDTRTXint.	ISS - an exception from an LDC or STC instruction	When AArch32 is supported
0b000111	Access to SME, SVE, Advanced SIMD or floating-point functionality trapped by CPACR_EL1.FPEN, CPTR_EL2.FPEN, CPTR_EL2.TFP, or CPTR_EL3.TFP control. Excludes exceptions resulting from CPACR_EL1 when the value of HCR_EL2.TGE is 1, or because SVE or Advanced SIMD and floating-point are not implemented. These are reported with EC value 0b000000 as described in 'The EC used to report an exception routed to EL2 because HCR_EL2.TGE is 1'.	ISS - an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps	
0b001001	Trapped use of a Pointer authentication instruction because HCR_EL2.API == 0 || SCR_EL3.API == 0.	ISS - an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 || SCR_EL3.API == 0	When FEAT_PAuth is implemented
0b001010	Trapped execution of an LD64B, ST64B, ST64BV, or ST64BV0 instruction.	ISS - an exception from an LD64B or ST64B* instruction	When FEAT_LS64 is implemented
0b001100	Trapped MRRC access with (coproc==0b1110).	ISS - an exception from an MCRR or MRRC access	When AArch32 is supported
0b001101	Branch Target Exception.	ISS - an exception from Branch Target Identification instruction	When FEAT_BTI is implemented
0b001110	Illegal Execution state.	ISS - an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b010011	SMC instruction execution in AArch32 state, when SMC is not disabled.	ISS - an exception from SMC instruction execution in AArch32 state	When AArch32 is supported
0b010101	SVC instruction execution in AArch64 state.	ISS - an exception from HVC or SVC instruction execution	When AArch64 is supported

Value	Meaning	Link	Applies
`0b010110`	HVC instruction execution in AArch64 state, when HVC is not disabled.	ISS - an exception from HVC or SVC instruction execution	When AArch64 is supported
`0b010111`	SMC instruction execution in AArch64 state, when SMC is not disabled.	ISS - an exception from SMC instruction execution in AArch64 state	When AArch64 is supported
`0b011000`	Trapped MSR, MRS or System instruction execution in AArch64 state, that is not reported using EC 0b000000, 0b000001 or 0b000111. This includes all instructions that cause exceptions that are part of the encoding space defined in 'System instruction class encoding overview', except for those exceptions reported using EC values 0b000000, 0b000001, or 0b000111.	ISS - an exception from MSR, MRS, or System instruction execution in AArch64 state	When AArch64 is supported
`0b011001`	Access to SVE functionality trapped as a result of CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ, that is not reported using EC 0b000000.	ISS - an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ	When FEAT_SVE is implemented
`0b011011`	Exception from an access to a TSTART instruction at EL0 when SCTLR_EL1.TME0 == 0, EL0 when SCTLR_EL2.TME0 == 0, at EL1 when SCTLR_EL1.TME == 0, at EL2 when SCTLR_EL2.TME == 0 or at EL3 when SCTLR_EL3.TME == 0.	ISS - an exception from a TSTART instruction	When FEAT_TME is implemented
`0b011100`	Exception from a Pointer Authentication instruction authentication failure	ISS - an exception from a Pointer Authentication instruction authentication failure	When FEAT_FPAC is implemented
`0b011101`	Access to SME functionality trapped as a result of CPACR_EL1.SMEN, CPTR_EL2.SMEN, CPTR_EL2.TSM, CPTR_EL3.ESM, or an attempted execution of an instruction that is illegal because of the value of PSTATE.SM or PSTATE.ZA, that is not reported using EC 0b000000.	ISS - an exception due to SME functionality	When FEAT_SME is implemented
`0b011110`	Exception from a Granule Protection Check	ISS - an exception from a Granule Protection Check	When FEAT_RME is implemented
`0b011111`	IMPLEMENTATION DEFINED exception to EL3.	ISS - an IMPLEMENTATION DEFINED exception to EL3	
`0b100000`	Instruction Abort from a lower Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from an Instruction Abort	
`0b100001`	Instruction Abort taken without a change in Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from an Instruction Abort	

Value	Meaning	Link	Applies
0b100010	PC alignment fault exception.	ISS - an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b100100	Data Abort from a lower Exception level. Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from a Data Abort	
0b100101	Data Abort taken without a change in Exception level. Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS - an exception from a Data Abort	
0b100110	SP alignment fault exception.	ISS - an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b101100	Trapped floating-point exception taken from AArch64 state. This EC value is valid if the implementation supports trapping of floating-point exceptions, otherwise it is reserved. Whether a floating-point implementation supports trapping of floating-point exceptions is IMPLEMENTATION DEFINED.	ISS - an exception from a trapped floating-point exception	When AArch64 is supported
0b101111	SError interrupt.	ISS - an SError interrupt	
0b111100	BRK instruction execution in AArch64 state. This is reported in ESR_EL3 only if a BRK instruction is executed in EL3. This is the only debug exception that can be taken to EL3 when EL3 is using AArch64.	ISS - an exception from execution of a Breakpoint instruction	When AArch64 is supported

All other EC values are reserved by Arm, and:

- Unused values in the range 0b000000 - 0b101100 (0x00 - 0x2C) are reserved for future use for synchronous exceptions.
- Unused values in the range 0b101101 - 0b111111 (0x2D - 0x3F) are reserved for future use, and might be used for synchronous or asynchronous exceptions.

The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [25]

Instruction Length for synchronous exceptions. Possible values of this bit are:

Value	Meaning
0b0	16-bit instruction trapped.

Value	Meaning
`0b1`	32-bit instruction trapped. This value is also used when the exception is one of the following: • An SError interrupt. • An Instruction Abort exception. • A PC alignment fault exception. • An SP alignment fault exception. • A Data Abort exception for which the value of the ISV bit is 0. • An Illegal Execution state exception. • Any debug exception except for Breakpoint instruction exceptions. • An exception reported using EC value 0b000000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS, bits [24:0]

Instruction Specific Syndrome. Architecturally, this field can be defined independently for each defined Exception class. However, in practice, some ISS encodings are used for more than one Exception class.

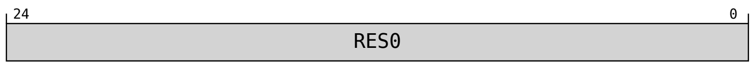
Typically, an ISS encoding has a number of subfields. When an ISS subfield holds a register number, the value returned in that field is the AArch64 view of the register number.

For an exception taken from AArch32 state, see 'Mapping of the general-purpose registers between the Execution states'.

If the AArch32 register descriptor is 0b1111, then:

- If the instruction that generated the exception was not UNPREDICTABLE, the field takes the value 0b11111.
- If the instruction that generated the exception was UNPREDICTABLE, the field takes an UNKNOWN value that must be either:
  - The AArch64 view of the register number of a register that might have been used at the Exception level from which the exception was taken.
  - The value 0b11111.

ISS encoding for exceptions with an unknown reason



Bits [24:0]

Reserved, RES0.

Additional information for exceptions with an unknown reason

When an exception is reported using this EC code the IL field is set to 1.

This EC code is used for all exceptions that are not covered by any other EC value. This includes exceptions that are generated in the following situations:

- The attempted execution of an instruction bit pattern that has no allocated instruction or that is not accessible at the current Exception level and Security state, including:

- A read access using a System register pattern that is not allocated for reads or that does not permit reads at the current Exception level and Security state.
- A write access using a System register pattern that is not allocated for writes or that does not permit writes at the current Exception level and Security state.
- Instruction encodings that are unallocated.
- Instruction encodings for instructions or System registers that are not implemented in the implementation.

- In Debug state, the attempted execution of an instruction bit pattern that is not accessible in Debug state.
- In Non-debug state, the attempted execution of an instruction bit pattern that is not accessible in Non-debug state.
- In AArch32 state, attempted execution of a short vector floating-point instruction.
- In an implementation that does not include Advanced SIMD and floating-point functionality, an attempted access to Advanced SIMD or floating-point functionality under conditions where that access would be permitted if that functionality was present. This includes the attempted execution of an Advanced SIMD or floating-point instruction, and attempted accesses to Advanced SIMD and floating-point System registers.
- An exception generated because of the value of one of the SCTLR_EL1.{ITD, SED, CP15BEN} control bits.
- Attempted execution of:
  - An HVC instruction when disabled by HCR_EL2.HCD or SCR_EL3.HCE.
  - An SMC instruction when disabled by SCR_EL3.SMD.
  - An HLT instruction when disabled by EDSCR.HDE.
- Attempted execution of an MSR or MRS instruction to access SP_EL0 when the value of SPSel.SP is 0.
- Attempted execution of an MSR or MRS instruction using a _EL12 register name when HCR_EL2.E2H == 0.
- Attempted execution, in Debug state, of:
  - A DCPS1 instruction when the value of HCR_EL2.TGE is 1 and EL2 is disabled or not implemented in the current Security state.
  - A DCPS2 instruction from EL1 or EL0 when EL2 is disabled or not implemented in the current Security state.
  - A DCPS3 instruction when the value of EDSCR.SDD is 1, or when EL3 is not implemented.
- When EL3 is using AArch64, attempted execution from Secure EL1 of an SRS instruction using R13_mon. See 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32'.
- In Debug state when the value of EDSCR.SDD is 1, the attempted execution at EL2, EL1, or EL0 of an instruction that is configured to trap to EL3.
- In AArch32 state, the attempted execution of an MRS (banked register) or an MSR (banked register) instruction to SPSR_mon, SP_mon, or LR_mon.
- An exception that is taken to EL2 because the value of HCR_EL2.TGE is 1 that, if the value of HCR_EL2.TGE was 0 would have been reported with an ESR_ELx.EC value of 0b000111.
- In Non-transactional state, attempted execution of a TCOMMIT instruction.

ISS encoding for an exception from a WF instruction*

24	23–20	19–10	9–5	4–3	2	1–0
CV	COND	RES0	RN	RES0	RV	TI

CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:10]

Reserved, RES0.

RN, bits [9:5]

When FEAT_WFxT is implemented:

Register Number. Indicates the register number supplied for a WFET or WFIT instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [4:3]

Reserved, RES0.

RV, bit [2]

When FEAT_WFxT is implemented:

Register field Valid.

If TI[1] == 1, then this field indicates whether RN holds a valid register number for the register argument to the trapped WFET or WFIT instruction.

Value	Meaning
0b0	Register field invalid.
0b1	Register field valid.

If TI[1] == 0, then this field is RES0.

This field is set to 1 on a trap on WFET or WFIT.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TI, bits [1:0]

Trapped instruction. Possible values of this bit are:

Value	Meaning	Applies
0b00	WFI trapped.	
0b01	WFE trapped.	
0b10	WFIT trapped.	When FEAT_WFxT is implemented
0b11	WFET trapped.	When FEAT_WFxT is implemented

When FEAT_WFxT is implemented, this is a two bit field as shown. Otherwise, bit[1] is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a WF* instruction

The following fields describe configuration settings for generating this exception:

- SCTLR_EL1.{nTWE, nTWI}.
- HCR_EL2.{TWE, TWI}.
- SCR_EL3.{TWE, TWI}.

ISS encoding for an exception from an MCR or MRC access

24	23–20	19–17	16–14	13–10	9–5	4–1	0
CV	COND	Opc2	Opc1	CRn	Rt	CRm	Direction

CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc2, bits [19:17]

The Opc2 value from the issued instruction.

For a trapped VMRS access, holds the value 0b000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [16:14]

The Opc1 value from the issued instruction.

For a trapped VMRS access, holds the value 0b111.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

For a trapped VMRS access, holds the reg field from the VMRS instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

For a trapped VMRS access, holds the value 0b0000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
`0b0`	Write to System register space. MCR instruction.
`0b1`	Read from System register space. MRC or VMRS instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an MCR or MRC access

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000011:

- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- PMUSERENR_EL0.{ER, CR, SW, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- AMUSERENR_EL0.EN, for accesses to Activity Monitors registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TTLB, for execution of TLB maintenance instructions at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.{TSW, TPC, TPU} for execution of cache maintenance instructions at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TACR, for accesses to the Auxiliary Control Register at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TIDCP, for accesses to lockdown, DMA, and TCM operations at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.{TID1, TID2, TID3}, for accesses to ID registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- CPTR_EL2.TCPAC, for accesses to CPACR_EL1 or CPACR using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HSTR_EL2.T<n>, for accesses to System registers using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- CNTHCTL_EL2.EL1PCEN, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- CPTR_EL3.TCPAC, for accesses to CPACR from EL1 and EL2, and accesses to HCPTR from EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- For information on other traps using EC value 0b000011, see 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32'.
- If FEAT_FGT is implemented, MCR or MRC access to some registers at EL0, trapped to EL2.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000101:

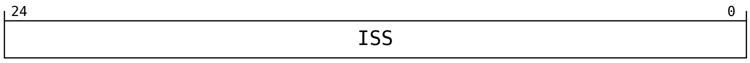
- CPACR_EL1.TTA for accesses to trace registers, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- MDSCR_EL1.TDCC, for accesses to the Debug Communications Channel (DCC) registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- If FEAT_FGT is implemented, MDCR_EL2.TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- HCR_EL2.TID0, for accesses to the JIDR register in the ID group 0 at EL0 and EL1 using AArch32, MRC access (coproc == 0b1110) trapped to EL2.

- CPTR_EL2.TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers DBGDRAR and DBGDSAR using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL2.TDOSA, for accesses to powerdown debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL2.TDA, for accesses to other debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- CPTR_EL3.TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.
- MDCR_EL3.TDOSA, for accesses to powerdown debug registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.
- MDCR_EL3.TDA, for accesses to other debug registers, using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001000:

- HCR_EL2.TID0, for accesses to the FPSID register in ID group 0 at EL1 using AArch32 state, VMRS access trapped to EL2.
- HCR_EL2.TID3, for accesses to registers in ID group 3 including MVFR0, MVFR1 and MVFR2, VMRS access trapped to EL2.

ISS encoding for an exception from an LD64B or ST64B instruction*

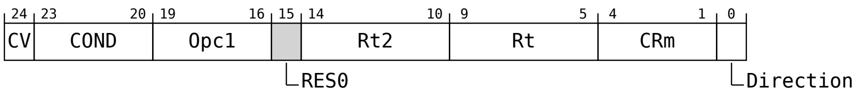


ISS, bits [24:0]

Value	Meaning
0b0000000000000000000000000	ST64BV instruction trapped.
0b0000000000000000000000001	ST64BV0 instruction trapped.
0b0000000000000000000000010	LD64B or ST64B instruction trapped.

All other values are reserved.

ISS encoding for an exception from an MCRR or MRRC access



CV, bit [24]

Condition code valid.

Value	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [19:16]

The Opc1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [15]

Reserved, RES0.

Rt2, bits [14:10]

The Rt2 value from the issued instruction, the second general-purpose register used for the transfer.

If the Rt2 value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt2 value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:

- The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
- The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the first general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
`0b0`	Write to System register space. MCRR instruction.
`0b1`	Read from System register space. MRRC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an MCRR or MRRC access

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000100:

- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.

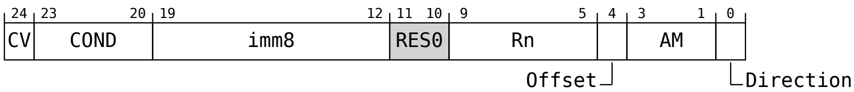
- PMUSERENR_EL0.{CR, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- AMUSERENR_EL0.{EN}, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- HSTR_EL2.T<n>, for accesses to System registers using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- CNTHCTL_EL2.{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- If FEAT_FGT is implemented, HDFGRTR_EL2.PMCCNTR_EL0 for MRRC access and HDFGWTR_EL2.PMCCNTR_EL0 for MCRR access to PMCCNTR at EL0, trapped to EL2.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001100:

- MDSCR_EL1.TDCC, for accesses to the Debug ROM registers DBGDSAR and DBGDRAR at EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers DBGDRAR and DBGDSAR using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- MDCR_EL3.TDA, for accesses to debug registers, using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.
- CPACR_EL1.TTA for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- CPTR_EL2.TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- CPTR_EL3.TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.

If the Armv8-A architecture is implemented with an ETMv4 implementation, MCRR and MRRC accesses to trace registers are UNDEFINED and the resulting exception is higher priority than an exception due to these traps.

ISS encoding for an exception from an LDC or STC instruction



CV, bit [24]

Condition code valid.

Value	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

imm8, bits [19:12]

The immediate value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:10]

Reserved, RES0.

Rn, bits [9:5]

The Rn value from the issued instruction, the general-purpose register used for the transfer.

If the Rn value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rn value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:

- The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
- The value 0b11111.

See ‘Mapping of the general-purpose registers between the Execution states’.

This field is valid only when AM[2] is 0, indicating an immediate form of the LDC or STC instruction. When AM[2] is 1, indicating a literal form of the LDC or STC instruction, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Offset, bit [4]

Indicates whether the offset is added or subtracted:

Value	Meaning
`0b0`	Subtract offset.
`0b1`	Add offset.

This bit corresponds to the U bit in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

AM, bits [3:1]

Addressing mode. The permitted values of this field are:

Value	Meaning
`0b000`	Immediate unindexed.
`0b001`	Immediate post-indexed.
`0b010`	Immediate offset.
`0b011`	Immediate pre-indexed.
`0b100`	For a trapped STC instruction or a trapped T32 LDC instruction this encoding is reserved.
`0b110`	For a trapped STC instruction, this encoding is reserved.

The values 0b101 and 0b111 are reserved. The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in ‘Reserved values in System and memory-mapped registers and translation table entries’.

Bit [2] in this subfield indicates the instruction form, immediate or literal.

Bits [1:0] in this subfield correspond to the bits {P, W} in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
0b0	Write to memory. STC instruction.
0b1	Read from memory. LDC instruction.

The reset behavior of this field is:

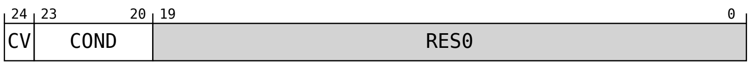
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an LDC or STC instruction

The following fields describe the configuration settings for the traps that are reported using EC value 0b000110:

- MDSCR_EL1.TDCC, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint trapped to EL1 or EL2.
- MDCR_EL2.TDA, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint MCR or MRC access trapped to EL2.
- MDCR_EL3.TDA, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint MCR or MRC access trapped to EL3.
- If FEAT_FGT is implemented, MDCR_EL2.TDCC for LDC and STC accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.

ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps



The accesses covered by this trap include:

- Execution of SVE or Advanced SIMD and floating-point instructions.
- Accesses to the Advanced SIMD and floating-point System registers.
- Execution of SME instructions.

For an implementation that does not include either SVE or support for Advanced SIMD and floating-point, the exception is reported using the EC value 0b000000.

CV, bit [24]

Condition code valid.

Value	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:0]

Reserved, RES0.

Additional information for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps

The following fields describe the configuration settings for the traps that are reported using EC value 0b000111:

- CPACR_EL1.FPEN, for accesses to SIMD and floating-point registers trapped to EL1.
- CPTR_EL2.FPEN and CPTR_EL2.TFP, for accesses to SIMD and floating-point registers trapped to EL2.
- CPTR_EL3.TFP, for accesses to SIMD and floating-point registers trapped to EL3.

ISS encoding for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ

24 — 0
RES0

The accesses covered by this trap include:

- Execution of SVE instructions when the PE is not in Streaming SVE mode.
- Accesses to the SVE System registers, ZCR_ELx.

For an implementation that does not include SVE, the exception is reported using the EC value 0b000000.

Bits [24:0]

Reserved, RES0.

Additional information for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ

The following fields describe the configuration settings for the traps that are reported using EC value 0b011001:

- CPACR_EL1.ZEN, for execution of SVE instructions and accesses to SVE registers at EL0 or EL1, trapped to EL1.
- CPTR_EL2.ZEN and CPTR_EL2.TZ, for execution of SVE instructions and accesses to SVE registers at EL0, EL1, or EL2, trapped to EL2.
- CPTR_EL3.EZ, for execution of SVE instructions and accesses to SVE registers from all Exception levels, trapped to EL3.

ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault

24 … 0
RES0

Bits [24:0]

Reserved, RES0.

Additional information for an exception from an Illegal Execution state, or a PC or SP alignment fault

There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about these exceptions, see 'The Illegal Execution state exception' and 'PC alignment checking'.

'SP alignment checking' describes the configuration settings for generating SP alignment fault exceptions.

ISS encoding for an exception from HVC or SVC instruction execution

24 … 16	15 … 0
RES0	imm16

Bits [24:16]

Reserved, RES0.

imm16, bits [15:0]

The value of the immediate field from the HVC or SVC instruction.

For an HVC instruction, and for an A64 SVC instruction, this is the value of the imm16 field of the issued instruction.

For an A32 or T32 SVC instruction:

- If the instruction is unconditional, then:
  - For the T32 instruction, this field is zero-extended from the imm8 field of the instruction.
  - For the A32 instruction, this field is the bottom 16 bits of the imm24 field of the instruction.
- If the instruction is conditional, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from HVC or SVC instruction execution

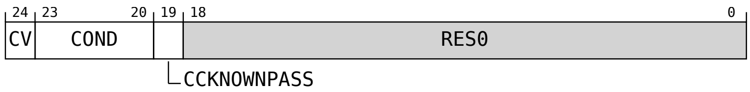
In AArch32 state, the HVC instruction is unconditional, and a conditional SVC instruction generates an exception only if it passes its condition code check. Therefore, the syndrome information for these exceptions does not require conditionality information.

For T32 and A32 instructions, see 'SVC' and 'HVC'.

For A64 instructions, see 'SVC' and 'HVC'.

If FEAT_FGT is implemented, HFGITR_EL2.{SVC_EL1, SVC_EL0} control fine-grained traps on SVC execution.

ISS encoding for an exception from SMC instruction execution in AArch32 state



For an SMC instruction that completes normally and generates an exception that is taken to EL3, the ISS encoding is RES0.

For an SMC instruction that is trapped to EL2 from EL1 because HCR_EL2.TSC is 1, the ISS encoding is as shown in the diagram.

CV, bit [24]

Condition code valid.

Value	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

This field is valid only if CCKNOWNPASS is 1, otherwise it is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.

   – CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.

- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

This field is valid only if CCKNOWNPASS is 1, otherwise it is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CCKNOWNPASS, bit [19]

Indicates whether the instruction might have failed its condition code check.

Value	Meaning
0b0	The instruction was unconditional, or was conditional and passed its condition code check.
0b1	The instruction was conditional, and might have failed its condition code check.

In an implementation in which an SMC instruction that fails it code check is not trapped, this field can always return the value 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [18:0]

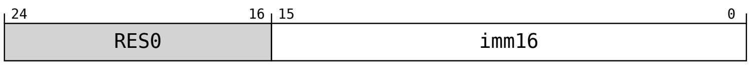
Reserved, RES0.

Additional information for an exception from SMC instruction execution in AArch32 state

HCR_EL2.TSC describes the configuration settings for trapping SMC instructions to EL2.

'System calls' describes the case where these exceptions are trapped to EL3.

ISS encoding for an exception from SMC instruction execution in AArch64 state



Bits [24:16]

Reserved, RES0.

imm16, bits [15:0]

The value of the immediate field from the issued SMC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from SMC instruction execution in AArch64 state

The value of ISS[24:0] described here is used both:

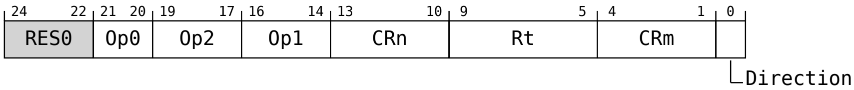
- When an SMC instruction is trapped from EL1 modes.

- When an SMC instruction is not trapped, so completes normally and generates an exception that is taken to EL3.

HCR_EL2.TSC describes the configuration settings for trapping SMC from EL1 modes.

'System calls' describes the case where these exceptions are trapped to EL3.

ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state



Bits [24:22]

Reserved, RES0.

Op0, bits [21:20]

The Op0 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op2, bits [19:17]

The Op2 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op1, bits [16:14]

The Op1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Value	Meaning
`0b0`	Write access, including MSR instructions.
`0b1`	Read access, including MRS instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from MSR, MRS, or System instruction execution in AArch64 state

For exceptions caused by System instructions, see 'System instructions' subsection of 'Branches, exception generating and System instructions' for the encoding values returned by an instruction.

The following fields describe configuration settings for generating the exception that is reported using EC value 0b011000:

- SCTLR_EL1.UCI, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- SCTLR_EL1.UCT, for accesses to CTR_EL0 using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- SCTLR_EL1.DZE, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- SCTLR_EL1.UMA, for accesses to the PSTATE interrupt masks using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- CPACR_EL1.TTA, for accesses to the trace registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- MDSCR_EL1.TDCC, for accesses to the Debug Communications Channel (DCC) registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- If FEAT_FGT is implemented, MDCR_EL2.TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN} accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- PMUSERENR_EL0.{ER, CR, SW, EN}, for accesses to the Performance Monitor registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- AMUSERENR_EL0.EN, for accesses to Activity Monitors registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TDZ, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TTLB, for execution of TLB maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{TSW, TPC, TPU}, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TACR, for accesses to the Auxiliary Control Register, ACTLR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TIDCP, for accesses to lockdown, DMA, and TCM operations using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{TID1, TID2, TID3}, for accesses to ID group 1, ID group 2 or ID group 3 registers, using AArch64 state, MSR or MRS access trapped to EL2.
- CPTR_EL2.TCPAC, for accesses to CPACR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.

- CPTR_EL2.TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TTRF, for accesses to the trace filter control register, TRFCR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDOSA, for accesses to powerdown debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- CNTHCTL_EL2.{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDA, for accesses to debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.APK, for accesses to Pointer authentication key registers. using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{NV, NV1}, for Nested virtualization register access, using AArch64 state, MSR or MRS access, trapped to EL2.
- HCR_EL2.AT, for execution of AT S1E* instructions, using AArch64 state, MSR or MRS access, trapped to EL2.
- HCR_EL2.{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access, trapped to EL2.
- SCR_EL3.APK, for accesses to Pointer authentication key registers, using AArch64 state, MSR or MRS access trapped to EL3.
- SCR_EL3.ST, for accesses to the Counter-timer Physical Secure timer registers, using AArch64 state, MSR or MRS access trapped to EL3.
- SCR_EL3.{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TCPAC, for accesses to CPTR_EL2 and CPACR_EL1 using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TTRF, for accesses to the trace filter control registers, TRFCR_EL1 and TRFCR_EL2, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TDA, for accesses to debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TDOSA, for accesses to powerdown debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access, trapped to EL3.
- If FEAT_EVT is implemented, the following registers control traps for EL1 and EL0 Cache controls that use this EC value:
  - HCR_EL2.{TTLBOS, TTLBIS, TICAB, TOCU, TID4}.
  - HCR2.{TTLBIS, TICAB, TOCU, TID4}.
- If FEAT_FGT is implemented:
  - SCR_EL3.FGTEn, for accesses to the fine-grained trap registers, MSR or MRS access at EL2 trapped to EL3.
  - HFGRTR_EL2 for reads and HFGWTR_EL2 for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 trapped to EL2.
  - HFGITR_EL2 for execution of system instructions, MSR or MRS access trapped to EL2
  - HDFGRTR_EL2 for reads and HDFGWTR_EL2 for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 state trapped to EL2.

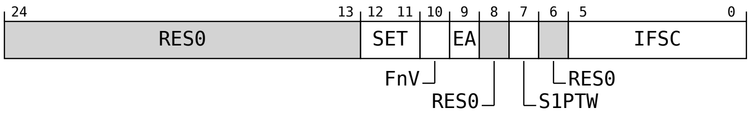
- HAFGRTR_EL2 for reads of Activity Monitor counters, using AArch64 state, MRS access at EL0 and EL1 trapped to EL2.
- If FEAT_RNG_TRAP is implemented:
  - SCR_EL3.TRNDR for reads of RNDR and RNDRRS using AArch64 state, MRS access trapped to EL3.
- If FEAT_SME is implemented:
  - CPTR_EL3.ESM, for MSR or MRS accesses to SMPRI_EL1 at EL1, EL2, and EL3, trapped to EL3.
  - CPTR_EL3.ESM, for MSR or MRS accesses to SMPRIMAP_EL2 at EL2 and EL3, trapped to EL3.
  - SCTLR_EL1.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, trapped to EL1 or EL2.
  - SCTLR_EL2.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, trapped to EL2.
  - SCR_EL3.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, EL1, and EL2, trapped to EL3.

ISS encoding for an IMPLEMENTATION DEFINED exception to EL3

IMPLEMENTATION DEFINED, bits [24:0]

IMPLEMENTATION DEFINED

ISS encoding for an exception from an Instruction Abort



Bits [24:13]

Reserved, RES0.

SET, bits [12:11]

When FEAT_RAS is implemented:

Synchronous Error Type. When IFSC is 0b010000, describes the PE error state after taking the Instruction Abort exception.

Value	Meaning
0b00	Recoverable state (UER).
0b10	Uncontainable (UC).
0b11	Restartable state (UEO).

All other values are reserved.

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

Value	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [8]

Reserved, RES0.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

Value	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

Value	Meaning	Applies
`0b000000`	Address size fault, level 0 of translation or translation table base register.	
`0b000001`	Address size fault, level 1.	
`0b000010`	Address size fault, level 2.	
`0b000011`	Address size fault, level 3.	
`0b000100`	Translation fault, level 0.	
`0b000101`	Translation fault, level 1.	
`0b000110`	Translation fault, level 2.	
`0b000111`	Translation fault, level 3.	
`0b001001`	Access flag fault, level 1.	
`0b001010`	Access flag fault, level 2.	
`0b001011`	Access flag fault, level 3.	
`0b001000`	Access flag fault, level 0.	When FEAT_LPA2 is implemented
`0b001100`	Permission fault, level 0.	When FEAT_LPA2 is implemented
`0b001101`	Permission fault, level 1.	
`0b001110`	Permission fault, level 2.	
`0b001111`	Permission fault, level 3.	
`0b010000`	Synchronous External abort, not on translation table walk or hardware update of translation table.	
`0b010011`	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
`0b010100`	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
`0b010101`	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
`0b010110`	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
`0b010111`	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
`0b011000`	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
`0b011011`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
`0b011100`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented

Value	Meaning	Applies
`0b011101`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
`0b011110`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
`0b011111`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
`0b100011`	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
`0b100100`	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
`0b100101`	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
`0b100110`	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
`0b100111`	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
`0b101000`	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
`0b101001`	Address size fault, level -1.	When FEAT_LPA2 is implemented
`0b101011`	Translation fault, level -1.	When FEAT_LPA2 is implemented
`0b110000`	TLB conflict abort.	
`0b110001`	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see ‘The level associated with MMU faults’.

Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception due to SME functionality

24:3	2:0
RES0	SMTC

The accesses covered by this trap include:

- Execution of SME instructions.
- Execution of SVE and Advanced SIMD instructions, when the PE is in Streaming SVE mode.
- Direct accesses of SVCR, SMCR_EL1, SMCR_EL2, SMCR_EL3.

Bits [24:3]

Reserved, RES0.

SMTC, bits [2:0]

SME Trap Code. Identifies the reason for instruction trapping.

Value	Meaning
0b000	Access to SME functionality trapped as a result of CPACR_EL1.SMEN, CPTR_EL2.SMEN, CPTR_EL2.TSM, or CPTR_EL3.ESM, that is not reported using EC 0b000000.
0b001	Advanced SIMD, SVE, or SVE2 instruction trapped because PSTATE.SM is 1.
0b010	SME instruction trapped because PSTATE.SM is 0.
0b011	SME instruction trapped because PSTATE.ZA is 0.

All other values are reserved.

Additional information for an exception due to SME functionality

The following fields describe the configuration settings for the traps that are reported using the EC value 0b011101:

- CPACR_EL1.SMEN, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR and SMCR_EL1 System registers at EL1 and EL0, trapped to EL1 or EL2.
- CPTR_EL2.SMEN and CPTR_EL2.TSM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR, SMCR_EL1, SMCR_EL2 at EL2, EL1, or EL0, trapped to EL2.
- CPTR_EL3.ESM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR, SMCR_EL1, SMCR_EL2, SMCR_EL3 from all Exception levels and any Security state, trapped to EL3.

ISS encoding for an exception from a Granule Protection Check

24:22	21	20	19:14	13	12:11	10:9	8	7	6	5:0
RES0	S2PTW	InD	GPCSC	VNCR	RES0	RES0	CM	S1PTW	WnR	xFSC

Bits [24:22]

Reserved, RES0.

S2PTW, bit [21]

Indicates whether the Granule Protection Check exception was on an access made for a stage 2 translation table walk.

Value	Meaning
`0b0`	Fault not on a stage 2 translation table walk.
`0b1`	Fault on a stage 2 translation table walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

InD, bit [20]

Indicates whether the Granule Protection Check exception was on an instruction or data access.

Value	Meaning
`0b0`	Data access.
`0b1`	Instruction access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GPCSC, bits [19:14]

Granule Protection Check Status Code.

Value	Meaning
`0b000000`	GPT address size fault at level 0.
`0b000001`	GPT address size fault at level 1.
`0b000100`	GPT walk fault at level 0.
`0b000101`	GPT walk fault at level 1.
`0b001100`	Granule protection fault at level 0.
`0b001101`	Granule protection fault at level 1.
`0b010100`	Synchronous External abort on GPT fetch at level 0.
`0b010101`	Synchronous External abort on GPT fetch at level 1.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [13]

When FEAT_NV2 is implemented

VNCR, bit [13]

Indicates that the fault came from use of VNCR_EL2 register by EL1 code.

Value	Meaning
0b0	The fault was not generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.
0b1	The fault was generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

When InD is '1', this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

Bit [13]

Reserved, RES0.

Bits [12:11]

Reserved, RES0.

Bits [10:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

Value	Meaning
0b0	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

Indicates whether the Granule Protection Check exception was on an access for stage 2 translation for a stage 1 translation table walk:

Value	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

Value	Meaning
0b0	Abort caused by an instruction reading from a memory location.
0b1	Abort caused by an instruction writing to a memory location.

When InD is '1', this field is RES0.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

xFSC, bits [5:0]

Instruction or Data Fault Status Code.

Value	Meaning	Applies
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented

Value	Meaning	Applies
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

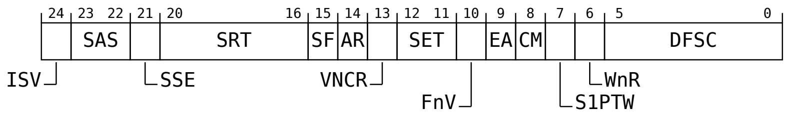
Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a Data Abort



When FEAT_LS64 is implemented, if a memory access generated by an ST64BV or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this ISS encoding includes ISS2, bits[36:32].

ISV, bit [24]

Instruction Syndrome Valid. Indicates whether the syndrome information in ISS[23:14] is valid.

Value	Meaning
0b0	No valid instruction syndrome. ISS[23:14] are RES0.
0b1	ISS[23:14] hold a valid instruction syndrome.

In ESR_EL2, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For other faults reported in ESR_EL2, ISV is 0 except for the following stage 2 aborts:

- AArch64 loads and stores of a single general-purpose register (including the register specified with 0b11111, including those with Acquire/Release semantics, but excluding Load Exclusive or Store Exclusive and excluding those with writeback).
- AArch32 instructions where the instruction:
  - Is an LDR, LDA, LDRT, LDRSH, LDRSHT, LDRH, LDAH, LDRHT, LDRSB, LDRSBT, LDRB, LDAB, LDRBT, STR, STL, STRT, STRH, STLH, STRHT, STRB, STLB, or STRBT instruction.

- Is not performing register writeback.
- Is not using R15 as a source or destination register.

For these stage 2 aborts, ISV is UNKNOWN if the exception was generated in Debug state in memory access mode, and otherwise indicates whether ISS[23:14] hold a valid syndrome.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault. ISV is 0 for all other faults reported in ESR_EL1 or ESR_EL3.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

For ISS reporting, a stage 2 abort on a stage 1 translation table walk does not return a valid instruction syndrome, and therefore ISV is 0 for these aborts.

When FEAT_RAS is not implemented, it is IMPLEMENTATION DEFINED whether ISV is set to 1 or 0 on a synchronous External abort on a stage 2 translation table walk.

When FEAT_MTE2 is implemented, for a synchronous Tag Check Fault abort taken to ELx, ESR_ELx.FNV is 0 and FAR_ELx is valid.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SAS, bits [23:22]

When ISV == 1:

Syndrome Access Size. Indicates the size of the access attempted by the faulting operation.

Value	Meaning
`0b00`	Byte
`0b01`	Halfword
`0b10`	Word
`0b11`	Doubleword

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SSE, bit [21]

When ISV == 1:

Syndrome Sign Extend. For a byte, halfword, or word load operation, indicates whether the data item must be sign extended.

Value	Meaning
`0b0`	Sign-extension not required.
`0b1`	Data item must be sign-extended.

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

For all other operations, this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SRT, bits [20:16]

When ISV == 1:

Syndrome Register Transfer. The register number of the Wt/Xt/Rt operand of the faulting instruction.

If the exception was taken from an Exception level that is using AArch32, then this is the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states'.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SF, bit [15]

When ISV == 1:

Width of the register accessed by the instruction is Sixty-Four.

Value	Meaning
`0b0`	Instruction loads/stores a 32-bit wide register.
`0b1`	Instruction loads/stores a 64-bit wide register.

This field specifies the register width identified by the instruction, not the Execution state.

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

AR, bit [14]

When ISV == 1:

Acquire/Release.

Value	Meaning
`0b0`	Instruction did not have acquire/release semantics.
`0b1`	Instruction did have acquire/release semantics.

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bit [13]

When FEAT_NV2 is implemented

VNCR, bit [13]

Indicates that the fault came from use of VNCR_EL2 register by EL1 code.

Value	Meaning
`0b0`	The fault was not generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.
`0b1`	The fault was generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

Bit [13]

Reserved, RES0.

Bits [12:11]

When FEAT_RAS is implemented and FEAT_LS64 is not implemented

SET, bits [12:11]

Synchronous Error Type. When DFSC is 0b010000, describes the PE error state after taking the Data Abort exception.

Value	Meaning
`0b00`	Recoverable state (UER).
`0b10`	Uncontainable (UC).
`0b11`	Restartable state (UEO).

All other values are reserved.

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

This field is valid only if the DFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_LS64 is implemented

LST, bits [12:11]

Load/Store Type. Used when an LD64B, ST64B, ST64BV, or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

Value	Meaning
`0b01`	An ST64BV instruction generated the Data Abort.
`0b10`	An LD64B or ST64B instruction generated the Data Abort.
`0b11`	An ST64BV0 instruction generated the Data Abort.

All other values are reserved.

This field is valid only if the DFSC code is 0b110101. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

Value	Meaning
`0b0`	FAR is valid.
`0b1`	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the DFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

Value	Meaning
`0b0`	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
`0b1`	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

Value	Meaning
`0b0`	Fault not on a stage 2 translation for a stage 1 translation table walk.
`0b1`	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

Value	Meaning
0b0	Abort caused by an instruction reading from a memory location.
0b1	Abort caused by an instruction writing to a memory location.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DFSC, bits [5:0]

Data Fault Status Code.

Value	Meaning	Applies
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	

Value	Meaning	Applies
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Synchronous Tag Check Fault.	When FEAT_MTE2 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100001	Alignment fault.	
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented

Value	Meaning	Applies
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
0b110100	IMPLEMENTATION DEFINED fault (Lockdown).	
0b110101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive or Atomic access).	

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

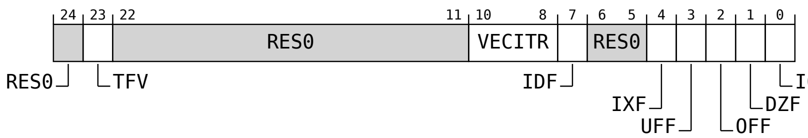
Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a trapped floating-point exception



Bit [24]

Reserved, RES0.

TFV, bit [23]

Trapped Fault Valid bit. Indicates whether the IDF, IXF, UFF, OFF, DZF, and IOF bits hold valid information about trapped floating-point exceptions.

Value	Meaning
0b0	The IDF, IXF, UFF, OFF, DZF, and IOF bits do not hold valid information about trapped floating-point exceptions and are UNKNOWN.

Value	Meaning
0b1	One or more floating-point exceptions occurred during an operation performed while executing the reported instruction. The IDF, IXF, UFF, OFF, DZF, and IOF bits indicate trapped floating-point exceptions that occurred. For more information, see 'Floating-point exceptions and exception traps'.

It is IMPLEMENTATION DEFINED whether this field is set to 0 on an exception generated by a trapped floating-point exception from an instruction that is performing floating-point operations on more than one lane of a vector.

This is not a requirement. Implementations can set this field to 1 on a trapped floating-point exception from an instruction and return valid information in the {IDF, IXF, UFF, OFF, DZF, IOF} fields.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [22:11]

Reserved, RES0.

VECITR, bits [10:8]

For a trapped floating-point exception from an instruction executed in AArch32 state this field is RES1.

For a trapped floating-point exception from an instruction executed in AArch64 state this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IDF, bit [7]

Input Denormal floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
0b0	Input denormal floating-point exception has not occurred.
0b1	Input denormal floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

IXF, bit [4]

Inexact floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
0b0	Inexact floating-point exception has not occurred.

Value	Meaning
`0b1`	Inexact floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

UFF, bit [3]

Underflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Underflow floating-point exception has not occurred.
`0b1`	Underflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

OFF, bit [2]

Overflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Overflow floating-point exception has not occurred.
`0b1`	Overflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DZF, bit [1]

Divide by Zero floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Divide by Zero floating-point exception has not occurred.
`0b1`	Divide by Zero floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IOF, bit [0]

Invalid Operation floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

Value	Meaning
`0b0`	Invalid Operation floating-point exception has not occurred.
`0b1`	Invalid Operation floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

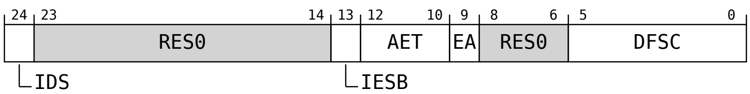
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a trapped floating-point exception

In an implementation that supports the trapping of floating-point exceptions:

- From an Exception level using AArch64, the FPCR.{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.
- From an Exception level using AArch32, the FPSCR.{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.

ISS encoding for an SError interrupt



IDS, bit [24]

IMPLEMENTATION DEFINED syndrome.

Value	Meaning
`0b0`	Bits [23:0] of the ISS field holds the fields described in this encoding. If FEAT_RAS is not implemented, bits [23:0] of the ISS field are RES0.
`0b1`	Bits [23:0] of the ISS field holds IMPLEMENTATION DEFINED syndrome information that can be used to provide additional information about the SError interrupt.

This field was previously called ISV.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [23:14]

Reserved, RES0.

IESB, bit [13]

When FEAT_IESB is implemented:

Implicit error synchronization event.

Value	Meaning
`0b0`	The SError interrupt was either not synchronized by the implicit error synchronization event or not taken immediately.
`0b1`	The SError interrupt was synchronized by the implicit error synchronization event and taken immediately.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

AET, bits [12:10]

When FEAT_RAS is implemented:

Asynchronous Error Type.

When DFSC is 0b010001, describes the PE error state after taking the SError interrupt exception.

Value	Meaning
`0b000`	Uncontainable (UC).
`0b001`	Unrecoverable state (UEU).
`0b010`	Restartable state (UEO).
`0b011`	Recoverable state (UER).
`0b110`	Corrected (CE).

All other values are reserved.

If multiple errors are taken as a single SError interrupt exception, the overall PE error state is reported.

Software can use this information to determine what recovery might be possible. The recovery software must also examine any implemented fault records to determine the location and extent of the error.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EA, bit [9]

When FEAT_RAS is implemented:

External abort type. When DFSC is 0b010001, provides an IMPLEMENTATION DEFINED classification of External aborts.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [8:6]

Reserved, RES0.

DFSC, bits [5:0]

When FEAT_RAS is implemented:

Data Fault Status Code.

Value	Meaning
0b000000	Uncategorized error.
0b010001	Asynchronous SError interrupt.

All other values are reserved.

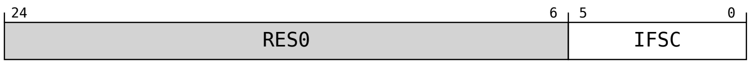
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

ISS encoding for an exception from a Breakpoint or Vector Catch debug exception



Bits [24:6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

Value	Meaning
0b100010	Debug exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

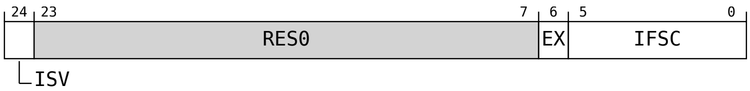
Additional information for an exception from a Breakpoint or Vector Catch debug exception

For more information about generating these exceptions:

- For exceptions from AArch64, see 'Breakpoint exceptions'.

- For exceptions from AArch32, see 'Breakpoint exceptions' and 'Vector Catch exceptions'.

ISS encoding for an exception from a Software Step exception



ISV, bit [24]

Instruction syndrome valid. Indicates whether the EX bit, ISS[6], is valid, as follows:

Value	Meaning
`0b0`	EX bit is RES0.
`0b1`	EX bit is valid.

See the EX bit description for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [23:7]

Reserved, RES0.

EX, bit [6]

Exclusive operation. If the ISV bit is set to 1, this bit indicates whether a Load-Exclusive instruction was stepped.

Value	Meaning
`0b0`	An instruction other than a Load-Exclusive instruction was stepped.
`0b1`	A Load-Exclusive instruction was stepped.

If the ISV bit is set to 0, this bit is RES0, indicating no syndrome data is available.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IFSC, bits [5:0]

Instruction Fault Status Code.

Value	Meaning
`0b100010`	Debug exception.

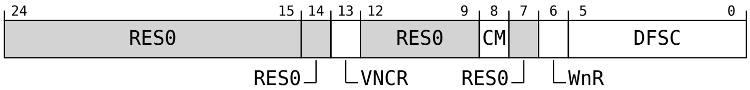
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a Software Step exception

For more information about generating these exceptions, see 'Software Step exceptions'.

ISS encoding for an exception from a Watchpoint exception



Bits [24:15]

Reserved, RES0.

Bit [14]

Reserved, RES0.

Bit [13]

When FEAT_NV2 is implemented

VNCR, bit [13]

Indicates that the watchpoint came from use of VNCR_EL2 register by EL1 code.

Value	Meaning
`0b0`	The watchpoint was not generated by the use of VNCR_EL2 by EL1 code.
`0b1`	The watchpoint was generated by the use of VNCR_EL2 by EL1 code.

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

Bit [13]

Reserved, RES0.

Bits [12:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Watchpoint exception came from a cache maintenance or address translation instruction:

Value	Meaning
`0b0`	The Watchpoint exception was not generated by the execution of one of the System instructions identified in the description of value 1.

Value	Meaning
`0b1`	The Watchpoint exception was generated by either the execution of a cache maintenance instruction or by a synchronous Watchpoint exception on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as a cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [7]

Reserved, RES0.

WnR, bit [6]

Write not Read. Indicates whether the Watchpoint exception was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

Value	Meaning
`0b0`	Watchpoint exception caused by an instruction reading from a memory location.
`0b1`	Watchpoint exception caused by an instruction writing to a memory location.

For Watchpoint exceptions on cache maintenance and address translation instructions, this bit always returns a value of 1.

For Watchpoint exceptions from an atomic instruction, this field is set to 0 if a read of the location would have generated the Watchpoint exception, otherwise it is set to 1.

If multiple watchpoints match on the same access, it is UNPREDICTABLE which watchpoint generates the Watchpoint exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DFSC, bits [5:0]

Data Fault Status Code.

Value	Meaning
`0b100010`	Debug exception.

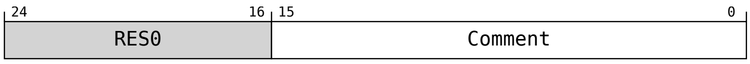
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a Watchpoint exception

For more information about generating these exceptions, see 'Watchpoint exceptions'.

ISS encoding for an exception from execution of a Breakpoint instruction



Bits [24:16]

Reserved, RES0.

Comment, bits [15:0]

Set to the instruction comment field value, zero extended as necessary.

For the AArch32 BKPT instructions, the comment field is described as the immediate field.

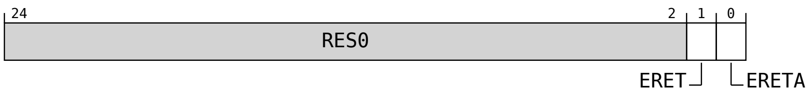
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from execution of a Breakpoint instruction

For more information about generating these exceptions, see 'Breakpoint instruction exceptions'.

ISS encoding for an exception from an ERET, ERETAA, or ERETAB instruction



This EC value applies when FEAT_FGT is implemented, or when HCR_EL2.NV is 1.

Bits [24:2]

Reserved, RES0.

ERET, bit [1]

Indicates whether an ERET or ERETA* instruction was trapped to EL2.

Value	Meaning
0b0	ERET instruction trapped to EL2.
0b1	ERETAA or ERETAB instruction trapped to EL2.

If this bit is 0, the ERETA field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ERETA, bit [0]

Indicates whether an ERETAA or ERETAB instruction was trapped to EL2.

Value	Meaning
0b0	ERETAA instruction trapped to EL2.

Value	Meaning
`0b1`	ERETAB instruction trapped to EL2.

When the ERET field is 0, this bit is RES0.

The reset behavior of this field is:

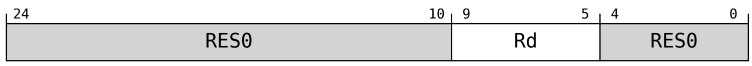
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from an ERET, ERETAA, or ERETAB instruction

For more information about generating these exceptions, see HCR_EL2.NV.

If FEAT_FGT is implemented, HFGITR_EL2.ERET controls fine-grained trap exceptions from ERET, ERETAA and ERETAB execution.

ISS encoding for an exception from a TSTART instruction



Bits [24:10]

Reserved, RES0.

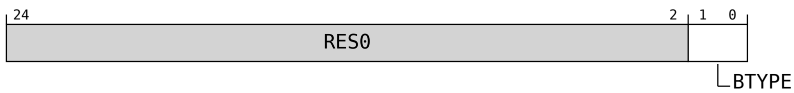
Rd, bits [9:5]

The Rd value from the issued instruction, the general purpose register used for the destination.

Bits [4:0]

Reserved, RES0.

ISS encoding for an exception from Branch Target Identification instruction



Bits [24:2]

Reserved, RES0.

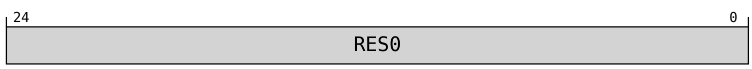
BTYPE, bits [1:0]

This field is set to the PSTATE.BTYPE value that generated the Branch Target Exception.

Additional information for an exception from Branch Target Identification instruction

For more information about generating these exceptions, see 'The AArch64 application level programmers model'.

ISS encoding for an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 || SCR_EL3.API == 0



Bits [24:0]

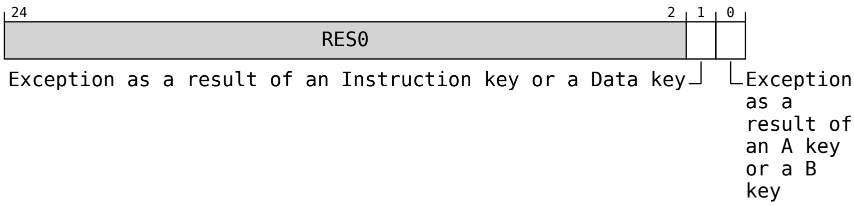
Reserved, RES0.

Additional information for an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 || SCR_EL3.API == 0

For more information about generating these exceptions, see:

- HCR_EL2.API, for exceptions from Pointer authentication instructions, using AArch64 state, trapped to EL2.
- SCR_EL3.API, for exceptions from Pointer authentication instructions, using AArch64 state, trapped to EL3.

ISS encoding for an exception from a Pointer Authentication instruction authentication failure



Bits [24:2]

Reserved, RES0.

Bit [1]

This field indicates whether the exception is as a result of an Instruction key or a Data key.

Value	Meaning
`0b0`	Instruction Key.
`0b1`	Data Key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [0]

This field indicates whether the exception is as a result of an A key or a B key.

Value	Meaning
`0b0`	A key.
`0b1`	B key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Additional information for an exception from a Pointer Authentication instruction authentication failure

The following instructions generate an exception when the Pointer Authentication Code (PAC) is incorrect:

- AUTIASP, AUTIAZ, AUTIA1716.
- AUTIBSP, AUTIBZ, AUTIB1716.

- AUTIA, AUTDA, AUTIB, AUTDB.
- AUTIZA, AUTIZB, AUTDZA, AUTDZB.

It is IMPLEMENTATION DEFINED whether the following instructions generate an exception directly from the authorization failure, rather than changing the address in a way that will generate a Translation fault when the address is accessed:

- RETAA, RETAB.
- BRAA, BRAB, BLRAA, BLRAB.
- BRAAZ, BRABZ, BLRAAZ, BLRABZ.
- ERETAA, ERETAB.
- LDRAA, LDRAB, whether the authenticated address is written back to the base register or not.

Accessing the ESR_EL3

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, ESR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return ESR_EL3;
```

MSR ESR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    ESR_EL3 = X[t];
```

A2.1.8 GPCCR_EL3, Granule Protection Check Control Register (EL3)

The GPCCR_EL3 characteristics are:

Purpose

The control register for Granule Protection Checks.

Attributes

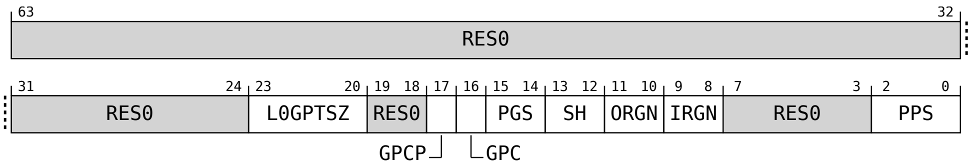
GPCCR_EL3 is a 64-bit register.

Configuration

This register is present only when FEAT_RME is implemented. Otherwise, direct accesses to GPCCR_EL3 are UNDEFINED.

Field descriptions

The GPCCR_EL3 bit assignments are:



Bits [63:24]

Reserved, RES0.

L0GPTSZ, bits [23:20]

Level 0 GPT entry size.

This field advertises the number of least-significant address bits protected by each entry in the level 0 GPT.

Value	Meaning
0b0000	30-bits. Each entry covers 1GB of address space.
0b0100	34-bits. Each entry covers 16GB of address space.
0b0110	36-bits. Each entry covers 64GB of address space.
0b1001	39-bits. Each entry covers 512GB of address space.

All other values are reserved.

Access to this field is **RO**.

Bits [19:18]

Reserved, RES0.

GPCP, bit [17]

Granule Protection Check Priority.

This control governs behavior of granule protection checks on fetches of stage 2 Table descriptors.

Value	Meaning
0b0	GPC faults are all reported with a priority that is consistent with the GPC being performed on any access to physical address space.
0b1	A GPC fault for the fetch of a Table descriptor for a stage 2 translation table walk might not be generated or reported. All other GPC faults are reported with a priority consistent with the GPC being performed on all accesses to physical address spaces.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GPC, bit [16]

Granule Protection Check Enable.

Value	Meaning
0b0	Granule protection checks are disabled. Accesses are not prevented by this mechanism.
0b1	All accesses to physical address spaces are subject to granule protection checks, except for fetches of GPT information and accesses governed by the GPCCR_EL3.GPCP control.

If any stage of translation is enabled, this bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0b0.

PGS, bits [15:14]

Physical Granule size.

Value	Meaning
0b00	4KB.
0b01	64KB.
0b10	16KB.

All other values are reserved.

The value of this field is permitted to be cached in a TLB.

Granule sizes not supported for stage 1 and not supported for stage 2, as defined in ID_AA64MMFR0_EL1, are reserved. For example, if ID_AA64MMFR0_EL1.TGran16 == 0b0000 and ID_AA64MMFR0_EL1.TGran16_2 == 0b0001, then the PGS encoding 0b10 is reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH, bits [13:12]

GPT fetch Shareability attribute

Value	Meaning
`0b00`	Non-shareable.
`0b10`	Outer Shareable.
`0b11`	Inner Shareable.

All other values are reserved.

Fetches of GPT information are made with the Shareability attribute that is configured in this field.

If both ORGN and IRGN are configured with Non-cacheable attributes, it is invalid to configure this field to any value other than 0b10.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ORGN, bits [11:10]

GPT fetch Outer cacheability attribute.

Value	Meaning
`0b00`	Normal memory, Outer Non-cacheable.
`0b01`	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.
`0b10`	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.
`0b11`	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.

Fetches of GPT information are made with the Outer cacheability attributes configured in this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IRGN, bits [9:8]

GPT fetch Inner cacheability attribute.

Value	Meaning
`0b00`	Normal memory, Inner Non-cacheable.

Value	Meaning
0b01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.

Fetches of GPT information are made with the Inner cacheability attributes configured in this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [7:3]

Reserved, RES0.

PPS, bits [2:0]

Protected Physical Address Size.

The size of the memory region protected by GPTBR_EL3, in terms of the number of least-significant address bits.

Value	Meaning
0b000	32 bits, 4GB protected address space.
0b001	36 bits, 64GB protected address space.
0b010	40 bits, 1TB protected address space.
0b011	42 bits, 4TB protected address space.
0b100	44 bits, 16TB protected address space.
0b101	48 bits, 256TB protected address space.
0b110	52 bits, 4PB protected address space.

All other values are reserved.

Configuration of this field to a value exceeding the implemented physical address size is invalid.

The value of this field is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing the GPCCR_EL3

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, GPCCR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0001	0b110

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return GPCCR_EL3;
```

MSR GPCCR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0001	0b110

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    GPCCR_EL3 = X[t];
```

A2.1.9 GPTBR_EL3, Granule Protection Table Base Register

The GPTBR_EL3 characteristics are:

Purpose

The control register for Granule Protection Table base address.

Attributes

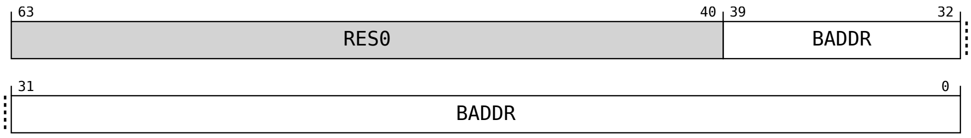
GPTBR_EL3 is a 64-bit register.

Configuration

This register is present only when FEAT_RME is implemented. Otherwise, direct accesses to GPTBR_EL3 are UNDEFINED.

Field descriptions

The GPTBR_EL3 bit assignments are:



Bits [63:40]

Reserved, RES0.

BADDR, bits [39:0]

Base address for the level 0 GPT.

This field represents bits [51:12] of the level 0 GPT base address.

The level 0 GPT is aligned in memory to the greater of:

- The size of the level 0 GPT in bytes.
- 4KB.

Bits [x:0] of the base address are treated as zero, where:

- x = Max(pps - l0gptsz + 2, 11)
- pps is derived from GPCCR_EL3.PPS as follows:

GPCCR_EL3.PPS	pps
0b000	32
0b001	36
0b010	40
0b011	42
0b100	44
0b101	48
0b110	52

- l0gptsz is derived from GPCCR_EL3.L0GPTSZ as follows:

GPCCR_EL3.L0GPTSZ	l0gptsz
0b0000	30
0b0100	34
0b0110	36
0b1001	39

If x is greater than 11, then BADDR[x - 12:0] are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing the GPTBR_EL3

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, GPTBR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0001	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return GPTBR_EL3;
```

MSR GPTBR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0001	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    GPTBR_EL3 = X[t];
```

A2.1.10 HCR_EL2, Hypervisor Configuration Register

The HCR_EL2 characteristics are:

Purpose

Provides configuration controls for virtualization, including defining whether various operations are trapped to EL2.

Attributes

HCR_EL2 is a 64-bit register.

Configuration

If EL2 is not implemented, this register is RES0 from EL3.

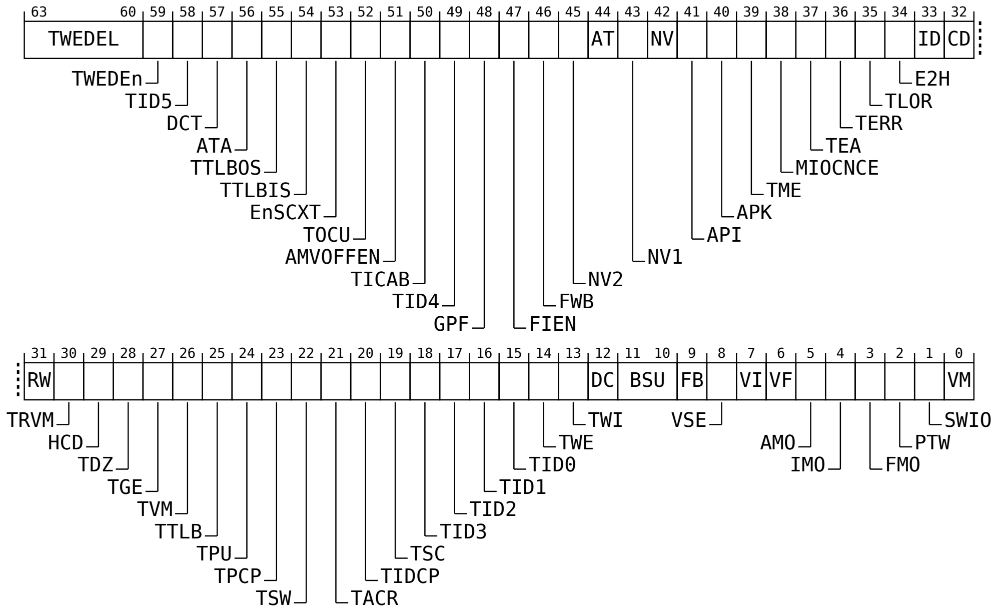
The bits in this register behave as if they are 0 for all purposes other than direct reads of the register if EL2 is not enabled in the current Security state.

AArch64 system register HCR_EL2 bits [31:0] are architecturally mapped to AArch32 system register HCR[31:0].

AArch64 system register HCR_EL2 bits [63:32] are architecturally mapped to AArch32 system register HCR2[31:0].

Field descriptions

The HCR_EL2 bit assignments are:



TWEDEL, bits [63:60]

When FEAT_TWED is implemented:

TWE Delay. A 4-bit unsigned number that, when HCR_EL2.TWEDEn is 1, encodes the minimum delay in taking a trap of WFE* caused by HCR_EL2.TWE as 2(TWEDEL + 8) cycles.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TWEDEn, bit [59]

When FEAT_TWED is implemented:

TWE Delay Enable. Enables a configurable delayed trap of the WFE* instruction caused by HCR_EL2.TWE.

Value	Meaning
`0b0`	The delay for taking the trap is IMPLEMENTATION DEFINED.
`0b1`	The delay for taking the trap is at least the number of cycles defined in HCR_EL2.TWEDEL.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TID5, bit [58]

When FEAT_MTE2 is implemented:

Trap ID group 5. Traps the following register accesses to EL2, when EL2 is enabled in the current Security state:

AArch64:

- GMID_EL1.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	The specified EL1 and EL0 accesses to ID group 5 registers are trapped to EL2.

When the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field has an Effective value of 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

DCT, bit [57]

When FEAT_MTE2 is implemented:

Default Cacheability Tagging. When HCR_EL2.DC is in effect, controls whether stage 1 translations are treated as Tagged or Untagged.

Value	Meaning
0b0	Stage 1 translations are treated as Untagged.
0b1	Stage 1 translations are treated as Tagged.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

ATA, bit [56]

When FEAT_MTE2 is implemented:

Allocation Tag Access. When HCR_EL2.{E2H,TGE} != {1,1}, controls EL1 and EL0 access to Allocation Tags.

Value	Meaning
0b0	Access to Allocation Tags is prevented. Accesses at EL1 to GCR_EL1, RGSR_EL1, TFSR_EL1, TFSR_EL2, or TFSRE0_EL1 that are not UNDEFINED are trapped to EL2.
0b1	This control does not prevent access to Allocation Tags.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TTLBOS, bit [55]

When FEAT_EVT is implemented:

Trap TLB maintenance instructions that operate on the Outer Shareable domain. Traps execution of those TLB maintenance instructions at EL1 to EL2, when EL2 is enabled in the current Security state. This applies to the following instructions:

TLBI VMALLE1OS, TLBI VAE1OS, TLBI ASIDE1OS,TLBI VAAE1OS, TLBI VALE1OS, TLBI VAALE1OS,TLBI RVAE1OS, TLBI RVAAE1OS,TLBI RVALE1OS, and TLBI RVAALE1OS.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Execution of the specified instructions are trapped to EL2.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TTLBIS, bit [54]

When FEAT_EVT is implemented:

Trap TLB maintenance instructions that operate on the Inner Shareable domain. Traps execution of those TLB maintenance instructions at EL1 to EL2, when EL2 is enabled in the current Security state. This applies to the following instructions:

- When EL1 is using AArch64, TLBI VMALLE1IS, TLBI VAE1IS, TLBI ASIDE1IS, TLBI VAAE1IS, TLBI VALE1IS, TLBI VAALE1IS, TLBI RVAE1IS, TLBI RVAAE1IS, TLBI RVALE1IS, and TLBI RVAALE1IS.
- When EL1 is using AArch32, TLBIALLIS, TLBIMVAIS, TLBIASIDIS, TLBIMVAAIS, TLBIMVALIS, and TLBIMVAALIS.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	Execution of the specified instructions are trapped to EL2.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EnSCXT, bit [53]

When FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented:

Enable Access to the SCXTNUM_EL1 and SCXTNUM_EL0 registers. The defined values are:

Value	Meaning
`0b0`	When HCR_EL2.E2H is 0 or HCR_EL2.TGE is 0, and EL2 is enabled in the current Security state, EL1 and EL0 access to SCXTNUM_EL0 and EL1 access to SCXTNUM_EL1 is disabled by this mechanism, causing an exception to EL2, and the values of these registers to be treated as 0. When HCR_EL2.{E2H, TGE} is {1, 1} and EL2 is enabled in the current Security state, EL0 access to SCXTNUM_EL0 is disabled by this mechanism, causing an exception to EL2, and the value of this register to be treated as 0.
`0b1`	This control does not cause accesses to SCXTNUM_EL0 or SCXTNUM_EL1 to be trapped.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1,1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TOCU, bit [52]

When FEAT_EVT is implemented:

Trap cache maintenance instructions that operate to the Point of Unification. Traps execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state. This applies to the following instructions:

- When SCTLR_EL1.UCI is 1, HCR_EL2.{TGE, E2H} is not {1, 1}, and EL0 is using AArch64, IC IVAU, DC CVAU.
- When EL1 is using AArch64, IC IVAU, IC IALLU, DC CVAU.
- When EL1 is using AArch32, ICIMVAU, ICIALLU, DCCMVAU.

An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition:

- IC IALLUIS and IC IALLU are always UNDEFINED at EL0 using AArch64.
- ICIMVAU, ICIALLU, ICIALLUIS, and DCCMVAU are always UNDEFINED at EL0 using AArch32.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	Execution of the specified instructions are trapped to EL2.

If the Point of Unification is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean by VA to the Point of Unification instruction can be trapped when the value of this control is 1.

If the Point of Unification is before any level of instruction cache, it is IMPLEMENTATION DEFINED whether the execution of any instruction cache invalidate to the Point of Unification instruction can be trapped when the value of this control is 1.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

AMVOFFEN, bit [51]

When FEAT_AMUv1p1 is implemented:

Activity Monitors Virtual Offsets Enable.

Value	Meaning
`0b0`	Virtualization of the Activity Monitors is disabled. Indirect reads of the virtual offset registers are zero.
`0b1`	Virtualization of the Activity Monitors is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TICAB, bit [50]

When FEAT_EVT is implemented:

Trap ICIALLUIS/IC IALLUIS cache maintenance instructions. Traps execution of those cache maintenance instructions at EL1 to EL2, when EL2 is enabled in the current Security state. This applies to the following instructions:

- When EL1 is using AArch64, IC IALLUIS.
- When EL1 is using AArch32, ICIALLUIS.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	EL1 execution of the specified instructions is trapped to EL2.

If the Point of Unification is before any level of instruction cache, it is IMPLEMENTATION DEFINED whether the execution of any instruction cache invalidate to the Point of Unification instruction can be trapped when the value of this control is 1.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TID4, bit [49]

When FEAT_EVT is implemented:

Trap ID group 4. Traps the following register accesses to EL2, when EL2 is enabled in the current Security state:

AArch64:

- EL1 reads of CCSIDR_EL1, CCSIDR2_EL1, CLIDR_EL1, and CSSELR_EL1.
- EL1 writes to CSSELR_EL1.

AArch32:

- EL1 reads of CCSIDR, CCSIDR2, CLIDR, and CSSELR.
- EL1 writes to CSSELR.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	The specified EL1 and EL0 accesses to ID group 4 registers are trapped to EL2.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

GPF, bit [48]

When FEAT_RME is implemented:

Controls the reporting of Granule protection faults at EL0 and EL1.

Value	Meaning
0b0	This control does not cause exceptions to be routed from EL0 and EL1 to EL2.
0b1	Instruction Aborts and Data Aborts due to GPFs from EL0 and EL1 are routed to EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

FIEN, bit [47]

When FEAT_RASv1p1 is implemented:

Fault Injection Enable. Unless this bit is set to 1, accesses to the ERXPFGCDN_EL1, ERXPFGCTL_EL1, and ERXPFGF_EL1 registers from EL1 generate a Trap exception to EL2, when EL2 is enabled in the current Security state, reported using EC syndrome value 0x18.

Value	Meaning
0b0	Accesses to the specified registers from EL1 are trapped to EL2, when EL2 is enabled in the current Security state.
0b1	This control does not cause any instructions to be trapped.

If EL2 is disabled in the current Security state, the Effective value of HCR_EL2.FIEN is 0b1.

If ERRIDR_EL1.NUM is zero, meaning no error records are implemented, or no error record accessible using System registers is owned by a node that implements the RAS Common Fault Injection Model Extension, then this bit might be RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

FWB, bit [46]

When FEAT_S2FWB is implemented:

Forced Write-Back. Defines the combined cacheability attributes in a 2 stage translation regime.

When FEAT_MTE2 is implemented, if the stage 1 page or block descriptor specifies the Tagged attribute, the final memory type is Tagged only if the final cacheable memory type is Inner and Outer Write-back cacheable and the final allocation hints are Read-Allocate, Write-Allocate.

Value	Meaning
`0b0`	When this bit is 0, then: • The combination of stage 1 and stage 2 translations on memory type and cacheability attributes are as described in the Armv8.0 architecture. For more information, see 'Combining the stage 1 and stage 2 attributes, EL1&0 translation regime'. • The encoding of the stage 2 memory type and cacheability attributes in bits[5:2] of the stage 2 page or block descriptors are as described in the Armv8.0 architecture.

Value	Meaning
`0b1`	When this bit is 1, then: • Bit[5] of stage 2 page or block descriptor is RES0. • When bit[4] of stage 2 page or block descriptor is 1 and when: – Bits[3:2] of stage 2 page or block descriptor are 0b11, the resultant memory type and inner or outer cacheability attribute is the same as the stage 1 memory type and inner or outer cacheability attribute. – Bits[3:2] of stage 2 page or block descriptor are 0b10, the resultant memory type and attribute is Normal Write-Back. – Bits[3:2] of stage 2 page or block descriptor are 0b0x, the resultant memory type will be Normal Non-cacheable except where the stage 1 memory type was Device-<attr> the resultant memory type will be Device-<attr> • When bit[4] of stage 2 page or block descriptor is 0 the memory type is Device, and when: – Bits[3:2] of stage 2 page or block descriptor are 0b00, the stage 2 memory type is Device-nGnRnE. – Bits[3:2] of stage 2 page or block descriptor are 0b01, the stage 2 memory type is Device-nGnRE. – Bits[3:2] of stage 2 page or block descriptor are 0b10, the stage 2 memory type is Device-nGRE. – Bits[3:2] of stage 2 page or block descriptor are 0b11, the stage 2 memory type is Device-GRE. • If the stage 1 translation specifies a cacheable memory type, then the stage 1 cache allocation hint is applied to the final cache allocation hint where the final memory type is cacheable. • If the stage 1 translation does not specify a cacheable memory type, then if the final memory type is cacheable, it is treated as read allocate, write allocate. The stage 1 and stage 2 memory types are combined in the manner described in 'Combining the stage 1 and stage 2 attributes, EL1&0 translation regime'.

In Secure state, this bit applies to both the Secure stage 2 translation and the Non-secure stage 2 translation.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NV2, bit [45]

When FEAT_NV2 is implemented:

Nested Virtualization. Changes the behaviors of HCR_EL2.{NV1, NV} to provide a mechanism for hardware to transform reads and writes from System registers into reads and writes from memory.

Value	Meaning
`0b0`	This bit has no effect on the behavior of HCR_EL2.{NV1, NV}. The behavior of HCR_EL2.{NV1, NV} is as defined for FEAT_NV.
`0b1`	Redefines behavior of HCR_EL2{NV1, NV} to enable: • Transformation of read/writes to registers into read/writes to memory. • Redirection of EL2 registers to EL1 registers. Any exception taken from EL1 and taken to EL1 causes SPSR_EL1.M[3:2] to be set to 0b10 and not 0b01.

When HCR_EL2.NV is 0, the Effective value of this field is 0 and this field is treated as 0 for all purposes other than direct reads and writes of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

AT, bit [44]

When FEAT_NV is implemented:

Address Translation. EL1 execution of the following address translation instructions is trapped to EL2, when EL2 is enabled in the current Security state, reported using EC syndrome value 0x18:

- AT S1E0R, AT S1E0W, AT S1E1R, AT S1E1W, AT S1E1RP, AT S1E1WP.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	EL1 execution of the specified instructions is trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bit [43]

When FEAT_NV2 is implemented

NV1, bit [43]

Nested Virtualization.

Value	Meaning
`0b0`	If HCR_EL2.{NV2, NV} are both 1, accesses executed from EL1 to implemented EL12, EL02, or EL2 registers are transformed to loads and stores. If HCR_EL2.NV2 is 0 or HCR_EL2.{NV2, NV} == {1, 0}, this control does not cause any instructions to be trapped.
`0b1`	If HCR_EL2.NV2 is 1, accesses executed from EL1 to implemented EL2 registers are transformed to loads and stores. If HCR_EL2.NV2 is 0, EL1 accesses to VBAR_EL1, ELR_EL1, SPSR_EL1, and, when FEAT_CSV2_2 or FEAT_CSV2_1p2 is implemented, SCXTNUM_EL1, are trapped to EL2, when EL2 is enabled in the current Security state, and are reported using EC syndrome value 0x18.

If HCR_EL2.NV2 is 1, the value of HCR_EL2.NV1 defines which EL1 register accesses are transformed to loads and stores. These transformed accesses have priority over the trapping of registers.

The trapping of EL1 registers caused by other control bits has priority over the transformation of these accesses.

If a register is specified that is not implemented by an implementation, then access to that register are UNDEFINED.

For the list of registers affected, see 'Enhanced support for nested virtualization'.

If HCR_EL2.{NV1, NV} is {0, 1}, any exception taken from EL1, and taken to EL1, causes the SPSR_EL1.M[3:2] to be set to 0b10, and not 0b01.

If HCR_EL2.{NV1, NV} is {1, 1}, then:

- The EL1 translation table Block and Page descriptors:
  - Bit[54] holds the PXN instead of the UXN.
  - Bit[53] is RES0.
  - Bit[6] is treated as 0 regardless of the actual value.
- If Hierarchical Permissions are enabled, the EL1 translation table Table descriptors are as follows:
  - Bit[61] is treated as 0 regardless of the actual value.
  - Bit[60] holds the PXNTable instead of the UXNTable.
  - Bit[59] is RES0.
- When executing at EL1, the PSTATE.PAN bit is treated as zero for all purposes except reading the value of the bit.
- When executing at EL1, the LDTR* instructions are treated as the equivalent LDR* instructions, and the STTR* instructions are treated as the equivalent STR* instructions.

If HCR_EL2.{NV1, NV} are {1, 0}, then the behavior is a CONSTRAINED UNPREDICTABLE choice of:

- Behaving as if HCR_EL2.NV is 1 and HCR_EL2.NV1 is 1 for all purposes other than reading back the value of the HCR_EL2.NV bit.
- Behaving as if HCR_EL2.NV is 0 and HCR_EL2.NV1 is 0 for all purposes other than reading back the value of the HCR_EL2.NV1 bit.

- Behaving with regard to the HCR_EL2.NV and HCR_EL2.NV1 bits behavior as defined in the rest of this description.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_NV is implemented

NV1, bit [43]

Nested Virtualization. EL1 accesses to certain registers are trapped to EL2, when EL2 is enabled in the current Security state.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 accesses to VBAR_EL1, ELR_EL1, SPSR_EL1, and, when FEAT_CSV2_2 or FEAT_CSV2_1p2 is implemented, SCXTNUM_EL1, are trapped to EL2, when EL2 is enabled in the current Security state, and are reported using EC syndrome value 0x18.

If HCR_EL2.NV is 1 and HCR_EL2.NV1 is 0, then the following effects also apply:

- Any exception taken from EL1, and taken to EL1, causes the SPSR_EL1.M[3:2] to be set to 0b10, and not 0b01.

If HCR_EL2.NV and HCR_EL2.NV1 are both set to 1, then the following effects also apply:

- The EL1 translation table Block and Page descriptors:
  - Bit[54] holds the PXN instead of the UXN.
  - Bit[53] is RES0.
  - Bit[6] is treated as 0 regardless of the actual value.
- If Hierarchical Permissions are enabled, the EL1 translation table Table descriptors are as follows:
  - Bit[61] is treated as 0 regardless of the actual value.
  - Bit[60] holds the PXNTable instead of the UXNTable.
  - Bit[59] is RES0.
- When executing at EL1, the PSTATE.PAN bit is treated as zero for all purposes except reading the value of the bit.
- When executing at EL1, the LDTR* instructions are treated as the equivalent LDR* instructions, and the STTR* instructions are treated as the equivalent STR* instructions.

If HCR_EL2.NV is 0 and HCR_EL2.NV1 is 1, then the behavior is a CONSTRAINED UNPREDICTABLE choice of:

- Behaving as if HCR_EL2.NV is 1 and HCR_EL2.NV1 is 1 for all purposes other than reading back the value of the HCR_EL2.NV bit.
- Behaving as if HCR_EL2.NV is 0 and HCR_EL2.NV1 is 0 for all purposes other than reading back the value of the HCR_EL2.NV1 bit.
- Behaving with regard to the HCR_EL2.NV and HCR_EL2.NV1 bits behavior as defined in the rest of this description.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bit [42]

When FEAT_NV2 is implemented

NV, bit [42]

Nested Virtualization.

When HCR_EL2.NV2 is 1, redefines register accesses so that:

- Instructions accessing the Special purpose registers SPSR_EL2 and ELR_EL2 instead access SPSR_EL1 and ELR_EL1 respectively.
- Instructions accessing the System registers ESR_EL2 and FAR_EL2 instead access ESR_EL1 and FAR_EL1.

When HCR_EL2.NV2 is 0, or if FEAT_NV2 is not implemented, traps functionality that is permitted at EL2 and would be UNDEFINED at EL1 if this field was 0, when EL2 is enabled in the current Security state. This applies to the following operations:

- EL1 accesses to Special-purpose registers that are not UNDEFINED at EL2.
- EL1 accesses to System registers that are not UNDEFINED at EL2.
- Execution of EL1 or EL2 translation regime address translation and TLB maintenance instructions for EL2 and above.

Value	Meaning
`0b0`	When this bit is set to 0, then the PE behaves as if HCR_EL2.NV2 is 0 for all purposes other than reading this register. This control does not cause any instructions to be trapped. When HCR_EL2.NV2 is 1, no FEAT_NV2 functionality is implemented.
`0b1`	When HCR_EL2.NV2 is 0, or if FEAT_NV2 is not implemented, EL1 accesses to the specified registers or the execution of the specified instructions are trapped to EL2, when EL2 is enabled in the current Security state. EL1 read accesses to the CurrentEL register return a value of 0x2. When HCR_EL2.NV2 is 1, this control redefines EL1 register accesses so that instructions accessing SPSR_EL2, ELR_EL2, ESR_EL2, and FAR_EL2 instead access SPSR_EL1, ELR_EL1, ESR_EL1, and FAR_EL1 respectively.

When HCR_EL2.NV2 is 0, or if FEAT_NV2 is not implemented, then:

- The System or Special-purpose registers for which accesses are trapped and reported using EC syndrome value 0x18 are as follows:
  - Registers accessed using MRS or MSR with a name ending in _EL2, except SP_EL2.
  - Registers accessed using MRS or MSR with a name ending in _EL12.
  - Registers accessed using MRS or MSR with a name ending in _EL02.
  - Special-purpose registers SPSR_irq, SPSR_abt, SPSR_und and SPSR_fiq, accessed using MRS or MSR.
  - Special-purpose register SP_EL1 accessed using the dedicated MRS or MSR instruction.
- The instructions for which the execution is trapped and reported using EC syndrome value 0x18 are as follows:

- EL2 translation regime Address Translation instructions and TLB maintenance instructions.
- EL1 translation regime Address Translation instructions and TLB maintenance instructions that are accessible only from EL2 and EL3.

- The instructions for which the execution is trapped as follows:
  - SMC in an implementation that does not include EL3 and when HCR_EL2.TSC is 1. HCR_EL2.TSC bit is not RES0 in this case. This is reported using EC syndrome value 0x17.
  - The ERET, ERETAA, and ERETAB instructions, reported using EC syndrome value 0x1A.

The priority of this trap is higher than the priority of the HCR_EL2.API trap. If both of these bits are set so that EL1 execution of an ERETAA or ERETAB instruction is trapped to EL2, then the syndrome reported is 0x1A.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_NV is implemented

NV, bit [42]

Nested Virtualization. Traps functionality that is permitted at EL2 and would be UNDEFINED at EL1 if this field was 0, when EL2 is enabled in the current Security state. This applies to the following operations:

- EL1 accesses to Special-purpose registers that are not UNDEFINED at EL2.
- EL1 accesses to System registers that are not UNDEFINED at EL2.
- Execution of EL1 or EL2 translation regime address translation and TLB maintenance instructions for EL2 and above.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 accesses to the specified registers or the execution of the specified instructions are trapped to EL2, when EL2 is enabled in the current Security state. EL1 read accesses to the CurrentEL register return a value of 0x2.

The System or Special-purpose registers for which accesses are trapped and reported using EC syndrome value 0x18 are as follows:

- Registers accessed using MRS or MSR with a name ending in _EL2, except SP_EL2.
- Registers accessed using MRS or MSR with a name ending in _EL12.
- Registers accessed using MRS or MSR with a name ending in _EL02.
- Special-purpose registers SPSR_irq, SPSR_abt, SPSR_und and SPSR_fiq, accessed using MRS or MSR.
- Special-purpose register SP_EL1 accessed using the dedicated MRS or MSR instruction.

The instructions for which the execution is trapped and reported using EC syndrome value 0x18 are as follows:

- EL2 translation regime Address Translation instructions and TLB maintenance instructions.
- EL1 translation regime Address Translation instructions and TLB maintenance instructions that are accessible only from EL2 and EL3.

The execution of the ERET, ERETAA, and ERETAB instructions are trapped and reported using EC syndrome value 0x1A.

The priority of this trap is higher than the priority of the HCR_EL2.API trap. If both of these bits are set so that EL1 execution of an ERETAA or ERETAB instruction is trapped to EL2, then the syndrome reported is 0x1A.

The execution of the SMC instructions in an implementation that does not include EL3 and when HCR_EL2.TSC is 1 are trapped and reported using EC syndrome value 0x17. HCR_EL2.TSC bit is not RES0 in this case.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

API, bit [41]

When FEAT_PAuth is implemented:

Controls the use of instructions related to Pointer Authentication:

- In EL0, when HCR_EL2.TGE==0 or HCR_EL2.E2H==0, and the associated SCTLR_EL1.En<N><M>==1.
- In EL1, the associated SCTLR_EL1.En<N><M>==1.

Traps are reported using EC syndrome value 0x09. The Pointer Authentication instructions trapped are:

- AUTDA, AUTDB, AUTDZA, AUTDZB, AUTIA, AUTIA1716, AUTIASP, AUTIAZ, AUTIB, AUTIB1716, AUTIBSP, AUTIBZ, AUTIZA, AUTIZB.
- PACGA, PACDA, PACDB, PACDZA, PACDZB, PACIA, PACIA1716, PACIASP, PACIAZ, PACIB, PACIB1716, PACIBSP, PACIBZ, PACIZA, PACIZB.
- RETAA, RETAB, BRAA, BRAB, BLRAA, BLRAB, BRAAZ, BRABZ, BLRAAZ, BLRABZ.
- ERETAA, ERETAB, LDRAA, and LDRAB.

Value	Meaning
`0b0`	The instructions related to Pointer Authentication are trapped to EL2, when EL2 is enabled in the current Security state and the instructions are enabled for the EL1&0 translation regime, from: • EL0 when HCR_EL2.TGE==0 or HCR_EL2.E2H==0. • EL1. If HCR_EL2.NV is 1, the HCR_EL2.NV trap takes precedence over the HCR_EL2.API trap for the ERETAA and ERETAB instructions. If EL2 is implemented and enabled in the current Security state and HFGITR_EL2.ERET == 1, execution at EL1 using AArch64 of ERETAA or ERETAB instructions is reported with EC syndrome value 0x1A with its associated ISS field, as the fine-grained trap has higher priority than the HCR_EL2.API == 0.
`0b1`	This control does not cause any instructions to be trapped.

If FEAT_PAuth is implemented but EL2 is not implemented or disabled in the current Security state, the system behaves as if this bit is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

APK, bit [40]

When FEAT_PAuth is implemented:

Trap registers holding "key" values for Pointer Authentication. Traps accesses to the following registers from EL1 to EL2, when EL2 is enabled in the current Security state, reported using EC syndrome value 0x18:

- APIAKeyLo_EL1, APIAKeyHi_EL1, APIBKeyLo_EL1, APIBKeyHi_EL1, APDAKeyLo_EL1, APDAKeyHi_EL1, APDBKeyLo_EL1, APDBKeyHi_EL1, APGAKeyLo_EL1, and APGAKeyHi_EL1.

Value	Meaning
`0b0`	Access to the registers holding "key" values for pointer authentication from EL1 are trapped to EL2, when EL2 is enabled in the current Security state.
`0b1`	This control does not cause any instructions to be trapped.

If FEAT_PAuth is implemented but EL2 is not implemented or is disabled in the current Security state, the system behaves as if this bit is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TME, bit [39]

When FEAT_TME is implemented:

Enables access to the TSTART, TCOMMIT, TTEST, and TCANCEL instructions at EL0 and EL1.

Value	Meaning
`0b0`	EL0 and EL1 accesses to TSTART, TCOMMIT, TTEST, and TCANCEL instructions are UNDEFINED.
`0b1`	This control does not cause any instruction to be UNDEFINED.

If EL2 is not implemented or is disabled in the current Security state, the Effective value of this bit is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

MIOCNCE, bit [38]

Mismatched Inner/Outer Cacheable Non-Coherency Enable, for the EL1&0 translation regimes.

Value	Meaning
0b0	For the EL1&0 translation regimes, for permitted accesses to a memory location that use a common definition of the Shareability and Cacheability of the location, there must be no loss of coherency if the Inner Cacheability attribute for those accesses differs from the Outer Cacheability attribute.
0b1	For the EL1&0 translation regimes, for permitted accesses to a memory location that use a common definition of the Shareability and Cacheability of the location, there might be a loss of coherency if the Inner Cacheability attribute for those accesses differs from the Outer Cacheability attribute.

For more information, see 'Mismatched memory attributes'.

This field can be implemented as RAZ/WI.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TEA, bit [37]

When FEAT_RAS is implemented:

Route synchronous External abort exceptions to EL2.

Value	Meaning
0b0	This control does not cause exceptions to be routed from EL0 and EL1 to EL2.
0b1	Route synchronous External abort exceptions from EL0 and EL1 to EL2, when EL2 is enabled in the current Security state, if not routed to EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TERR, bit [36]

When FEAT_RAS is implemented:

Trap Error record accesses. Trap accesses to the RAS error registers from EL1 to EL2 as follows:

- If EL1 is using AArch64 state, accesses to the following registers are trapped to EL2, reported using EC syndrome value 0x18:
  - ERRIDR_EL1, ERRSELR_EL1, ERXADDR_EL1, ERXCTLR_EL1, ERXFR_EL1, ERXMISC0_EL1, ERXMISC1_EL1, and ERXSTATUS_EL1.

  - When FEAT_RASv1p1 is implemented, ERXMISC2_EL1, and ERXMISC3_EL1.
- If EL1 is using AArch32 state, MCR or MRC accesses are trapped to EL2, reported using EC syndrome value 0x03, MCRR or MRRC accesses are trapped to EL2, reported using EC syndrome value 0x04:
  - ERRIDR, ERRSELR, ERXADDR, ERXADDR2, ERXCTLR, ERXCTLR2, ERXFR, ERXFR2, ERXMISC0, ERXMISC1, ERXMISC2, ERXMISC3, and ERXSTATUS.
  - When FEAT_RASv1p1 is implemented, ERXMISC4, ERXMISC5, ERXMISC6, and ERXMISC7.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	Accesses to the specified registers from EL1 generate a Trap exception to EL2, when EL2 is enabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TLOR, bit [35]

When FEAT_LOR is implemented:

Trap LOR registers. Traps Non-secure EL1 accesses to LORSA_EL1, LOREA_EL1, LORN_EL1, LORC_EL1, and LORID_EL1 registers to EL2.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	Non-secure EL1 accesses to the LOR registers are trapped to EL2.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

E2H, bit [34]

When FEAT_VHE is implemented:

EL2 Host. Enables a configuration where a Host Operating System is running in EL2, and the Host Operating System's applications are running in EL0.

Value	Meaning
0b0	The facilities to support a Host Operating System at EL2 are disabled.
0b1	The facilities to support a Host Operating System at EL2 are enabled.

For information on the behavior of this bit see 'Behavior of HCR_EL2.E2H'.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

ID, bit [33]

Stage 2 Instruction access cacheability disable. For the EL1&0 translation regime, when EL2 is enabled in the current Security state and HCR_EL2.VM==1, this control forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable.

Value	Meaning
0b0	This control has no effect on stage 2 of the EL1&0 translation regime.
0b1	Forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable.

This bit has no effect on the EL2, EL2&0, or EL3 translation regimes.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CD, bit [32]

Stage 2 Data access cacheability disable. For the EL1&0 translation regime, when EL2 is enabled in the current Security state and HCR_EL2.VM==1, this control forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable.

Value	Meaning
0b0	This control has no effect on stage 2 of the EL1&0 translation regime for data accesses and translation table walks.
0b1	Forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable.

This bit has no effect on the EL2, EL2&0, or EL3 translation regimes.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

RW, bit [31]

When EL1 is capable of using AArch32:

Execution state control for lower Exception levels:

Value	Meaning
0b0	Lower levels are all AArch32.
0b1	The Execution state for EL1 is AArch64. The Execution state for EL0 is determined by the current value of PSTATE.nRW when executing at EL0.

In an implementation that includes EL3, when EL2 is not enabled in Secure state, the PE behaves as if this bit has the same value as the SCR_EL3.RW bit for all purposes other than a direct read or write access of HCR_EL2.

The RW bit is permitted to be cached in a TLB.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 1 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RAO/WI

TRVM, bit [30]

Trap Reads of Virtual Memory controls. Traps EL1 reads of the virtual memory control registers to EL2, when EL2 is enabled in the current Security state, as follows:

- If EL1 is using AArch64 state, the following registers are trapped to EL2 and reported using EC syndrome value 0x18.
  - SCTLR_EL1, TTBR0_EL1, TTBR1_EL1, TCR_EL1, ESR_EL1, FAR_EL1, AFSR0_EL1, AFSR1_EL1, MAIR_EL1, AMAIR_EL1, CONTEXTIDR_EL1.
- If EL1 is using AArch32 state, accesses using MRC to the following registers are trapped to EL2 and reported using EC syndrome value 0x03, accesses using MRRC are trapped to EL2 and reported using EC syndrome value 0x04:
  - SCTLR, TTBR0, TTBR1, TTBCR, TTBCR2, DACR, DFSR, IFSR, DFAR, IFAR, ADFSR, AIFSR, PRRR, NMRR, MAIR0, MAIR1, AMAIR0, AMAIR1, CONTEXTIDR.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.

Value	Meaning
0b1	EL1 read accesses to the specified Virtual Memory controls are trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

EL2 provides a second stage of address translation, that a hypervisor can use to remap the address map defined by a Guest OS. In addition, a hypervisor can trap attempts by a Guest OS to write to the registers that control the memory system. A hypervisor might use this trap as part of its virtualization of memory management.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

HCD, bit [29]

When EL3 is not implemented:

HVC instruction disable. Disables EL1 execution of HVC instructions, from both Execution states, when EL2 is enabled in the current Security state, reported using EC syndrome value 0x00.

Value	Meaning
0b0	HVC instruction execution is enabled at EL2 and EL1.
0b1	HVC instructions are UNDEFINED at EL2 and EL1. Any resulting exception is taken to the Exception level at which the HVC instruction is executed.

HVC instructions are always UNDEFINED at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TDZ, bit [28]

Trap DC ZVA instructions. Traps EL0 and EL1 execution of DC ZVA instructions to EL2, when EL2 is enabled in the current Security state, from AArch64 state only, reported using EC syndrome value 0x18.

If FEAT_MTE is implemented, this trap also applies to DC GVA and DC GZVA.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.

Value	Meaning
0b1	In AArch64 state, any attempt to execute an instruction this trap applies to at EL1, or at EL0 when the instruction is not UNDEFINED at EL0, is trapped to EL2 when EL2 is enabled in the current Security state. Reading the DCZID_EL0 returns a value that indicates that the instructions this trap applies to are not supported.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TGE, bit [27]

Trap General Exceptions, from EL0.

Value	Meaning
0b0	This control has no effect on execution at EL0.
0b1	When EL2 is not enabled in the current Security state, this control has no effect on execution at EL0. When EL2 is enabled in the current Security state, in all cases: • All exceptions that would be routed to EL1 are routed to EL2. • If EL1 is using AArch64, the SCTLR_EL1.M field is treated as being 0 for all purposes other than returning the result of a direct read of SCTLR_EL1. • If EL1 is using AArch32, the SCTLR.M field is treated as being 0 for all purposes other than returning the result of a direct read of SCTLR. • All virtual interrupts are disabled. • Any IMPLEMENTATION DEFINED mechanisms for signaling virtual interrupts are disabled. • An exception return to EL1 is treated as an illegal exception return. • The MDCR_EL2.{TDRA, TDOSA, TDA, TDE} fields are treated as being 1 for all purposes other than returning the result of a direct read of MDCR_EL2. In addition, when EL2 is enabled in the current Security state, if: • HCR_EL2.E2H is 0, the Effective values of the HCR_EL2.{FMO, IMO, AMO} fields are 1. • HCR_EL2.E2H is 1, the Effective values of the HCR_EL2.{FMO, IMO, AMO} fields are 0. For further information on the behavior of this bit when E2H is 1, see 'Behavior of HCR_EL2.E2H'.

HCR_EL2.TGE must not be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TVM, bit [26]

Trap Virtual Memory controls. Traps EL1 writes to the virtual memory control registers to EL2, when EL2 is enabled in the current Security state, as follows:

- If EL1 is using AArch64 state, the following registers are trapped to EL2 and reported using EC syndrome value 0x18:
  - SCTLR_EL1, TTBR0_EL1, TTBR1_EL1, TCR_EL1, ESR_EL1, FAR_EL1, AFSR0_EL1, AFSR1_EL1, MAIR_EL1, AMAIR_EL1, CONTEXTIDR_EL1.
- If EL1 is using AArch32 state, accesses using MCR to the following registers are trapped to EL2 and reported using EC syndrome value 0x03, accesses using MCRR are trapped to EL2 and reported using EC syndrome value 0x04:
  - SCTLR, TTBR0, TTBR1, TTBCR, TTBCR2, DACR, DFSR, IFSR, DFAR, IFAR, ADFSR, AIFSR, PRRR, NMRR, MAIR0, MAIR1, AMAIR0, AMAIR1, CONTEXTIDR.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 write accesses to the specified EL1 virtual memory control registers are trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TTLB, bit [25]

Trap TLB maintenance instructions. Traps EL1 execution of TLB maintenance instructions to EL2, when EL2 is enabled in the current Security state, as follows:

- When EL1 is using AArch64 state, the following instructions are trapped to EL2 and reported using EC syndrome value 0x18:
  - TLBI VMALLE1, TLBI VAE1, TLBI ASIDE1, TLBI VAAE1, TLBI VALE1, TLBI VAALE1.
  - TLBI VMALLE1IS, TLBI VAE1IS, TLBI ASIDE1IS, TLBI VAAE1IS, TLBI VALE1IS, TLBI VAALE1IS.
  - If FEAT_TLBIOS is implemented, this trap applies to TLBI VMALLE1OS, TLBI VAE1OS, TLBI ASIDE1OS, TLBI VAAE1OS, TLBI VALE1OS, TLBI VAALE1OS.
  - If FEAT_TLBIRANGE is implemented, this trap applies to TLBI RVAE1, TLBI RVAAE1, TLBI RVALE1, TLBI RVAALE1, TLBI RVAE1IS, TLBI RVAAE1IS, TLBI RVALE1IS, TLBI RVAALE1IS.
  - If FEAT_TLBIOS and FEAT_TLBIRANGE are implemented, this trap applies to TLBI RVAE1OS, TLBI RVAAE1OS, TLBI RVALE1OS, TLBI RVAALE1OS.
- When EL1 is using AArch32 state, the following instructions are trapped to EL2 and reported using EC syndrome value 0x03:
  - TLBIALLIS, TLBIMVAIS, TLBIASIDIS, TLBIMVAAIS, TLBIMVALIS, TLBIMVAALIS.
  - TLBIALL, TLBIMVA, TLBIASID, TLBIMVAA, TLBIMVAL, TLBIMVAAL
  - ITLBIALL, ITLBIMVA, ITLBIASID.

– DTLBIALL, DTLBIMVA, DTLBIASID.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 execution of the specified TLB maintenance instructions are trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The TLB maintenance instructions are UNDEFINED at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TPU, bit [24]

Trap cache maintenance instructions that operate to the Point of Unification. Traps execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state as follows:

- If EL0 is using AArch64 state and the value of SCTLR_EL1.UCI is not 0, the following instructions are trapped to EL2 and reported with EC syndrome value 0x18:
  – IC IVAU, DC CVAU. If the value of SCTLR_EL1.UCI is 0 these instructions are UNDEFINED at EL0 and any resulting exception is higher priority than this trap to EL2.
- If EL1 is using AArch64 state, the following instructions are trapped to EL2 and reported with EC syndrome value 0x18:
  – IC IVAU, IC IALLU, IC IALLUIS, DC CVAU.
- If EL1 is using AArch32 state, the following instructions are trapped to EL2 and reported with EC syndrome value 0x18:
  – ICIMVAU, ICIALLU, ICIALLUIS, DCCMVAU.

An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition:

- IC IALLUIS and IC IALLU are always UNDEFINED at EL0 using AArch64.
- ICIMVAU, ICIALLU, ICIALLUIS, and DCCMVAU are always UNDEFINED at EL0 using AArch32.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Execution of the specified instructions is trapped to EL2, when EL2 is enabled in the current Security state.

If the Point of Unification is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean by VA to the Point of Unification instruction can be trapped when the value of this control is 1.

If the Point of Unification is before any level of instruction cache, it is IMPLEMENTATION DEFINED whether the execution of any instruction cache invalidate to the Point of Unification instruction can be trapped when the value of this control is 1.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [23]

When FEAT_DPB is implemented

TPCP, bit [23]

Trap data or unified cache maintenance instructions that operate to the Point of Coherency or Persistence. Traps execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state as follows:

- If EL0 is using AArch64 state and the value of SCTLR_EL1.UCI is not 0, the following instructions are trapped to EL2 and reported using EC syndrome value 0x18:
  - DC CIVAC, DC CVAC, DC CVAP. If the value of SCTLR_EL1.UCI is 0 these instructions are UNDEFINED at EL0 and any resulting exception is higher priority than this trap to EL2.
- If EL1 is using AArch64 state, the following instructions are trapped to EL2 and reported using EC syndrome value 0x18:
  - DC IVAC, DC CIVAC, DC CVAC, DC CVAP.
- If EL1 is using AArch32 state, the following instructions are trapped to EL2 and reported using EC syndrome value 0x03:
  - DCIMVAC, DCCIMVAC, DCCMVAC.

If FEAT_DPB2 is implemented, this trap also applies to DC CVADP.

If FEAT_MTE is implemented, this trap also applies to DC CIGVAC, DC CIGDVAC, DC IGVAC, DC IGDVAC, DC CGVAC, DC CGDVAC, DC CGVAP and DC CGDVAP.

If FEAT_DPB2 and FEAT_MTE are implemented, this trap also applies to DC CGVADP and DC CGDVADP.

- An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition:
  - AArch64 instructions which invalidate by VA to the Point of Coherency are always UNDEFINED at EL0 using AArch64.
  - DCIMVAC, DCCIMVAC, and DCCMVAC are always UNDEFINED at EL0 using AArch32.
- In Armv8.0 and Armv8.1, this field is named TPC. From Armv8.2, it is named TPCP.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	Execution of the specified instructions is trapped to EL2, when EL2 is enabled in the current Security state.

If the Point of Coherency is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean, invalidate, or clean and invalidate instruction that operates by VA to the point of coherency can be trapped when the value of this control is 1.

If HCR_EL2.{E2H, TGE} is set to {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

TPC, bit [23]

Trap data or unified cache maintenance instructions that operate to the Point of Coherency. Traps execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state as follows:

- If EL0 is using AArch64 state and the value of SCTLR_EL1.UCI is not 0, accesses to the following registers are trapped and reported using EC syndrome value 0x18:
  - DC CIVAC, DC CVAC. However, if the value of SCTLR_EL1.UCI is 0 these instructions are UNDEFINED at EL0 and any resulting exception is higher priority than this trap to EL2.
- If EL1 is using AArch64 state, accesses to DC IVAC, DC CIVAC, DC CVAC are trapped and reported using EC syndrome value 0x18.
- When EL1 is using AArch32, accesses to DCIMVAC, DCCIMVAC, and DCCMVAC are trapped and reported using EC syndrome value 0x03.
- An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition:
  - AArch64 instructions which invalidate by VA to the Point of Coherency are always UNDEFINED at EL0 using AArch64.
  - DCIMVAC, DCCIMVAC, and DCCMVAC are always UNDEFINED at EL0 using AArch32.
- In Armv8.0 and Armv8.1, this field is named TPC. From Armv8.2, it is named TPCP.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Execution of the specified instructions is trapped to EL2, when EL2 is enabled in the current Security state.

If the Point of Coherency is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean, invalidate, or clean and invalidate instruction that operates by VA to the point of coherency can be trapped when the value of this control is 1.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TSW, bit [22]

Trap data or unified cache maintenance instructions that operate by Set/Way. Traps execution of those cache maintenance instructions at EL1 to EL2, when EL2 is enabled in the current Security state as follows:

- If EL1 is using AArch64 state, accesses to DC ISW, DC CSW, DC CISW are trapped to EL2, reported using EC syndrome value 0x18.
- If EL1 is using AArch32 state, accesses to DCISW, DCCSW, DCCISW are trapped to EL2, reported using EC syndrome value 0x03.

If FEAT_MTE2 is implemented, this trap also applies to DC IGSW, DC IGDSW, DC CGSW, DC CGDW, DC CIGSW, and DC CIGDSW.

An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2, and these instructions are always UNDEFINED at EL0.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Execution of the specified instructions is trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TACR, bit [21]

Trap Auxiliary Control Registers. Traps EL1 accesses to the Auxiliary Control Registers to EL2, when EL2 is enabled in the current Security state, as follows:

- If EL1 is using AArch64 state, accesses to ACTLR_EL1 to EL2, are trapped to EL2 and reported using EC syndrome value 0x18.
- If EL1 is using AArch32 state, accesses to ACTLR and, if implemented, ACTLR2 are trapped to EL2 and reported using EC syndrome value 0x03.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 accesses to the specified registers are trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

ACTLR_EL1 is not accessible at EL0.

ACTLR and ACTLR2 are not accessible at EL0.

The Auxiliary Control Registers are IMPLEMENTATION DEFINED registers that might implement global control bits for the PE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TIDCP, bit [20]

Trap IMPLEMENTATION DEFINED functionality. Traps EL1 accesses to the encodings reserved for IMPLEMENTATION DEFINED functionality to EL2, when EL2 is enabled in the current Security state as follows:

- In AArch64 state, access to any of the encodings in the following reserved encoding spaces are trapped and reported using EC syndrome 0x18:
  - IMPLEMENTATION DEFINED System instructions, which are accessed using SYS and SYSL, with CRn == {11, 15}.
  - IMPLEMENTATION DEFINED System registers, which are accessed using MRS and MSR with the rS3_<op1>_<Cn>_<Cm>_<op2> register name.
- In AArch32 state, MCR and MRC access to instructions with the following encodings are trapped and reported using EC syndrome 0x03:

- All coproc==p15, CRn==c9, opc1 == {0-7}, CRm == {c0-c2, c5-c8}, opc2 == {0-7}.
- All coproc==p15, CRn==c10, opc1 =={0-7}, CRm == {c0, c1, c4, c8}, opc2 == {0-7}.
- All coproc==p15, CRn==c11, opc1=={0-7}, CRm == {c0-c8, c15}, opc2 == {0-7}.

When the value of HCR_EL2.TIDCP is 1, it is IMPLEMENTATION DEFINED whether any of this functionality accessed from EL0 is trapped to EL2. If it is not, then it is UNDEFINED, and any attempt to access it from EL0 generates an exception that is taken to EL1.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	EL1 accesses to or execution of the specified encodings reserved for IMPLEMENTATION DEFINED functionality are trapped to EL2, when EL2 is enabled in the current Security state.

An implementation can also include IMPLEMENTATION DEFINED registers that provide additional controls, to give finer-grained control of the trapping of IMPLEMENTATION DEFINED features.

Arm expects the trapping of EL0 accesses to these functions to EL2 to be unusual, and used only when the hypervisor is virtualizing EL0 operation. Arm strongly recommends that unless the hypervisor must virtualize EL0 operation, an EL0 access to any of these functions is UNDEFINED, as it would be if the implementation did not include EL2. The PE then takes any resulting exception to EL1.

The trapping of accesses to these registers from EL1 is higher priority than an exception resulting from the register access being UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TSC, bit [19]

Trap SMC instructions. Traps EL1 execution of SMC instructions to EL2, when EL2 is enabled in the current Security state.

If execution is in AArch64 state, the trap is reported using EC syndrome value 0x17.

If execution is in AArch32 state, the trap is reported using EC syndrome value 0x13.

HCR_EL2.TSC traps execution of the SMC instruction. It is not a routing control for the SMC exception. Trap exceptions and SMC exceptions have different preferred return addresses.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.

Value	Meaning
0b1	If EL3 is implemented, then any attempt to execute an SMC instruction at EL1 is trapped to EL2, when EL2 is enabled in the current Security state, regardless of the value of SCR_EL3.SMD. If EL3 is not implemented, FEAT_NV is implemented, and HCR_EL2.NV is 1, then any attempt to execute an SMC instruction at EL1 using AArch64 is trapped to EL2, when EL2 is enabled in the current Security state. If EL3 is not implemented, and either FEAT_NV is not implemented or HCR_EL2.NV is 0, then it is IMPLEMENTATION DEFINED whether: • Any attempt to execute an SMC instruction at EL1 is trapped to EL2, when EL2 is enabled in the current Security state. • Any attempt to execute an SMC instruction is UNDEFINED.

In AArch32 state, the Armv8-A architecture permits, but does not require, this trap to apply to conditional SMC instructions that fail their condition code check, in the same way as with traps on other conditional instructions.

SMC instructions are UNDEFINED at EL0.

If EL3 is not implemented, and either FEAT_NV is not implemented or HCR_EL2.NV is 0, then it is IMPLEMENTATION DEFINED whether this bit is:

- RES0.
- Implemented with the functionality as described in HCR_EL2.TSC.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TID3, bit [18]

Trap ID group 3. Traps EL1 reads of group 3 ID registers to EL2, when EL2 is enabled in the current Security state, as follows:

In AArch64 state:

- Reads of the following registers are trapped to EL2, reported using EC syndrome value 0x18:
  - ID_PFR0_EL1, ID_PFR1_EL1, ID_PFR2_EL1, ID_DFR0_EL1, ID_AFR0_EL1, ID_MMFR0_EL1, ID_MMFR1_EL1, ID_MMFR2_EL1, ID_MMFR3_EL1, ID_ISAR0_EL1, ID_ISAR1_EL1, ID_ISAR2_EL1, ID_ISAR3_EL1, ID_ISAR4_EL1, ID_ISAR5_EL1, MVFR0_EL1, MVFR1_EL1, MVFR2_EL1.
  - ID_AA64PFR0_EL1, ID_AA64PFR1_EL1, ID_AA64DFR0_EL1, ID_AA64DFR1_EL1, ID_AA64ISAR0_EL1, ID_AA64ISAR1_EL1, ID_AA64MMFR0_EL1, ID_AA64MMFR1_EL1, ID_AA64AFR0_EL1, ID_AA64AFR1_EL1.
  - If FEAT_FGT is implemented:
    - ID_MMFR4_EL1 and ID_MMFR5_EL1 are trapped to EL2.

    * ID_AA64MMFR2_EL1 and ID_ISAR6_EL1 are trapped to EL2.
    * ID_DFR1_EL1 is trapped to EL2.
    * ID_AA64ZFR0_EL1 is trapped to EL2.
    * ID_AA64SMFR0_EL1 is trapped to EL2.
    * ID_AA64ISAR2_EL1 is trapped to EL2.
    * This field traps all MRS accesses to registers in the following range that are not already mentioned in this field description: Op0 == 3, op1 == 0, CRn == c0, CRm == {c1-c7}, op2 == {0-7}.
  - If FEAT_FGT is not implemented:
    * ID_MMFR4_EL1 and ID_MMFR5_EL1 are trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to ID_MMFR4_EL1 or ID_MMFR5_EL1 are trapped to EL2.
    * ID_AA64MMFR2_EL1 and ID_ISAR6_EL1 are trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to ID_AA64MMFR2_EL1 or ID_ISAR6_EL1 are trapped to EL2.
    * ID_DFR1_EL1 is trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to ID_DFR1_EL1 are trapped to EL2.
    * ID_AA64ZFR0_EL1 is trapped to EL2, unless implemented as RAZ then it is IMPLEMENTATION DEFINED whether accesses to ID_AA64ZFR0_EL1 are trapped to EL2.
    * ID_AA64SMFR0_EL1 is trapped to EL2, unless implemented as RAZ then it is IMPLEMENTATION DEFINED whether accesses to ID_AA64SMFR0_EL1 are trapped to EL2.
    * ID_AA64ISAR2_EL1 is trapped to EL2, unless implemented as RAZ then it is IMPLEMENTATION DEFINED whether accesses to ID_AA64ISAR2_EL1 are trapped to EL2.
    * Otherwise, it is IMPLEMENTATION DEFINED whether this bit traps MRS accesses to registers in the following range that are not already mentioned in this field description: Op0 == 3, op1 == 0, CRn == c0, CRm == {c1-c7}, op2 == {0-7}.

In AArch32 state:

- VMRS access to MVFR0, MVFR1, and MVFR2, are trapped to EL2, reported using EC syndrome value 0x08, unless access is also trapped by HCPTR which takes priority.
- MRC access to the following registers are trapped to EL2, reported using EC syndrome value 0x03:
  - ID_PFR0, ID_PFR1, ID_PFR2, ID_DFR0, ID_AFR0, ID_MMFR0, ID_MMFR1, ID_MMFR2, ID_MMFR3, ID_ISAR0, ID_ISAR1, ID_ISAR2, ID_ISAR3, ID_ISAR4, ID_ISAR5.
  - If FEAT_FGT is implemented:
    * ID_MMFR4 and ID_MMFR5 are trapped to EL2.
    * ID_ISAR6 is trapped to EL2.
    * ID_DFR1 is trapped to EL2.
    * This field traps all MRC accesses to encodings in the following range that are not already mentioned in this field description: coproc == p15, opc1 == 0, CRn == c0, CRm == {c2-c7}, opc2 == {0-7}.
  - If FEAT_FGT is not implemented:
    * ID_MMFR4 and ID_MMFR5 are trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to ID_MMFR4 or ID_MMFR5 are trapped.
    * ID_ISAR6 is trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to ID_ISAR6 are trapped to EL2.

  * ID_DFR1 is trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to ID_DFR1 are trapped to EL2.
  * Otherwise, it is IMPLEMENTATION DEFINED whether this bit traps all MRC accesses to registers in the following range not already mentioned in this field description with coproc == p15, opc1 == 0, CRn == c0, CRm == {c2-c7}, opc2 == {0-7}.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	The specified EL1 read accesses to ID group 3 registers are trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TID2, bit [17]

Trap ID group 2. Traps the following register accesses to EL2, when EL2 is enabled in the current Security state, as follows:

- If EL1 is using AArch64, reads of CTR_EL0, CCSIDR_EL1, CCSIDR2_EL1, CLIDR_EL1, and CSSELR_EL1 are trapped to EL2, reported using EC syndrome value 0x18.
- If EL0 is using AArch64 and the value of SCTLR_EL1.UCT is not 0, reads of CTR_EL0 are trapped to EL2, reported using EC syndrome value 0x18. If the value of SCTLR_EL1.UCT is 0, then EL0 reads of CTR_EL0 are trapped to EL1 and the resulting exception takes precedence over this trap.
- If EL1 is using AArch64, writes to CSSELR_EL1 are trapped to EL2, reported using EC syndrome value 0x18.
- If EL1 is using AArch32, reads of CTR, CCSIDR, CCSIDR2, CLIDR, and CSSELR are trapped to EL2, reported using EC syndrome value 0x03.
- If EL1 is using AArch32, writes to CSSELR are trapped to EL2, reported using EC syndrome value 0x03.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	The specified EL1 and EL0 accesses to ID group 2 registers are trapped to EL2, when EL2 is enabled in the current Security state.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TID1, bit [16]

Trap ID group 1. Traps EL1 reads of the following registers to EL2, when EL2 is enabled in the current Security state as follows:

- In AArch64 state, accesses of REVIDR_EL1, AIDR_EL1, SMIDR_EL1, reported using EC syndrome value 0x18.
- In AArch32 state, accesses of TCMTR, TLBTR, REVIDR, AIDR, reported using EC syndrome value 0x03.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	The specified EL1 read accesses to ID group 1 registers are trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TID0, bit [15]

When AArch32 is supported:

Trap ID group 0. Traps the following register accesses to EL2:

- EL1 reads of the JIDR, reported using EC syndrome value 0x05.
- If the JIDR is RAZ from EL0, EL0 reads of the JIDR, reported using EC syndrome value 0x05.
- EL1 accesses using VMRS of the FPSID, reported using EC syndrome value 0x08.
- It is IMPLEMENTATION DEFINED whether the JIDR is RAZ or UNDEFINED at EL0. If it is UNDEFINED at EL0, then any resulting exception takes precedence over this trap.
- The FPSID is not accessible at EL0 using AArch32.
- Writes to the FPSID are ignored, and not trapped by this control.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	The specified EL1 read accesses to ID group 0 registers are trapped to EL2, when EL2 is enabled in the current Security state.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TWE, bit [14]

Traps EL0 and EL1 execution of WFE instructions to EL2, when EL2 is enabled in the current Security state, from both Execution states, reported using EC syndrome value 0x01.

When FEAT_WFxT is implemented, this trap also applies to the WFET instruction.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	Any attempt to execute a WFE instruction at EL0 or EL1 is trapped to EL2, when EL2 is enabled in the current Security state, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by SCTLR.nTWE or SCTLR_EL1.nTWE.

In AArch32 state, the attempted execution of a conditional WFE instruction is trapped only if the instruction passes its condition code check.

Since a WFE can complete at any time, even without a Wakeup event, the traps on WFE are not guaranteed to be taken, even if the WFE is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

For more information about when WFE instructions can cause the PE to enter a low-power state, see 'Wait for Event mechanism and Send event'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TWI, bit [13]

Traps EL0 and EL1 execution of WFI instructions to EL2, when EL2 is enabled in the current Security state, from both Execution states, reported using EC syndrome value 0x01.

When FEAT_WFxT is implemented, this trap also applies to the WFIT instruction.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	Any attempt to execute a WFI instruction at EL0 or EL1 is trapped to EL2, when EL2 is enabled in the current Security state, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by SCTLR.nTWI or SCTLR_EL1.nTWI.

In AArch32 state, the attempted execution of a conditional WFI instruction is trapped only if the instruction passes its condition code check.

Since a WFI can complete at any time, even without a Wakeup event, the traps on WFI are not guaranteed to be taken, even if the WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

For more information about when WFI instructions can cause the PE to enter a low-power state, see 'Wait for Interrupt'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DC, bit [12]

Default Cacheability.

Value	Meaning
0b0	This control has no effect on the EL1&0 translation regime.
0b1	In any Security state: • When EL1 is using AArch64, the PE behaves as if the value of the SCTLR_EL1.M field is 0 for all purposes other than returning the value of a direct read of SCTLR_EL1. • When EL1 is using AArch32, the PE behaves as if the value of the SCTLR.M field is 0 for all purposes other than returning the value of a direct read of SCTLR. • The PE behaves as if the value of the HCR_EL2.VM field is 1 for all purposes other than returning the value of a direct read of HCR_EL2. • The memory type produced by stage 1 of the EL1&0 translation regime is Normal Non-Shareable, Inner Write-Back Read-Allocate Write-Allocate, Outer Write-Back Read-Allocate Write-Allocate.

This field has no effect on the EL2, EL2&0, and EL3 translation regimes.

This bit is permitted to be cached in a TLB.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

BSU, bits [11:10]

Barrier Shareability upgrade. This field determines the minimum shareability domain that is applied to any barrier instruction executed from EL1 or EL0:

Value	Meaning
0b00	No effect.
0b01	Inner Shareable.
0b10	Outer Shareable.

Value	Meaning
0b11	Full system.

This value is combined with the specified level of the barrier held in its instruction, using the same principles as combining the shareability attributes from two stages of address translation.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0b00 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FB, bit [9]

Force broadcast. Causes the following instructions to be broadcast within the Inner Shareable domain when executed from EL1:

AArch32: BPIALL, TLBIALL, TLBIMVA, TLBIASID, DTLBIALL, DTLBIMVA, DTLBIASID, ITLBIALL, ITLBIMVA, ITLBIASID, TLBIMVAA, ICIALLU, TLBIMVAL, TLBIMVAAL.

AArch64: TLBI VMALLE1, TLBI VAE1, TLBI ASIDE1, TLBI VAAE1, TLBI VALE1, TLBI VAALE1, IC IALLU, TLBI RVAE1, TLBI RVAAE1, TLBI RVALE1, TLBI RVAALE1.

Value	Meaning
0b0	This field has no effect on the operation of the specified instructions.
0b1	When one of the specified instruction is executed at EL1, the instruction is broadcast within the Inner Shareable shareability domain.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VSE, bit [8]

Virtual SError interrupt.

Value	Meaning
0b0	This mechanism is not making a virtual SError interrupt pending.
0b1	A virtual SError interrupt is pending because of this mechanism.

The virtual SError interrupt is enabled only when the value of HCR_EL2.{TGE, AMO} is {0, 1}.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VI, bit [7]

Virtual IRQ Interrupt.

Value	Meaning
0b0	This mechanism is not making a virtual IRQ pending.
0b1	A virtual IRQ is pending because of this mechanism.

The virtual IRQ is enabled only when the value of HCR_EL2.{TGE, IMO} is {0, 1}.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VF, bit [6]

Virtual FIQ Interrupt.

Value	Meaning
0b0	This mechanism is not making a virtual FIQ pending.
0b1	A virtual FIQ is pending because of this mechanism.

The virtual FIQ is enabled only when the value of HCR_EL2.{TGE, FMO} is {0, 1}.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

AMO, bit [5]

Physical SError interrupt routing.

Value	Meaning
0b0	When executing at Exception levels below EL2, and EL2 is enabled in the current Security state: • When the value of HCR_EL2.TGE is 0, Physical SError interrupts are not taken to EL2. • When the value of HCR_EL2.TGE is 1, Physical SError interrupts are taken to EL2 unless they are routed to EL3. • Virtual SError interrupts are disabled.

Value	Meaning
`0b1`	When executing at any Exception level, and EL2 is enabled in the current Security state: • Physical SError interrupts are taken to EL2, unless they are routed to EL3. • When the value of HCR_EL2.TGE is 0, then virtual SError interrupts are enabled.

If EL2 is enabled in the current Security state and the value of HCR_EL2.TGE is 1:

- Regardless of the value of the AMO bit physical asynchronous External aborts and SError interrupts target EL2 unless they are routed to EL3.
- When FEAT_VHE is not implemented, or if HCR_EL2.E2H is 0, this field behaves as 1 for all purposes other than a direct read of the value of this bit.
- When FEAT_VHE is implemented and HCR_EL2.E2H is 1, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

For more information, see 'Asynchronous exception routing'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IMO, bit [4]

Physical IRQ Routing.

Value	Meaning
`0b0`	When executing at Exception levels below EL2, and EL2 is enabled in the current Security state: • When the value of HCR_EL2.TGE is 0, Physical IRQ interrupts are not taken to EL2. • When the value of HCR_EL2.TGE is 1, Physical IRQ interrupts are taken to EL2 unless they are routed to EL3. • Virtual IRQ interrupts are disabled.
`0b1`	When executing at any Exception level, and EL2 is enabled in the current Security state: • Physical IRQ interrupts are taken to EL2, unless they are routed to EL3. • When the value of HCR_EL2.TGE is 0, then Virtual IRQ interrupts are enabled.

If EL2 is enabled in the current Security state, and the value of HCR_EL2.TGE is 1:

- Regardless of the value of the IMO bit, physical IRQ Interrupts target EL2 unless they are routed to EL3.
- When FEAT_VHE is not implemented, or if HCR_EL2.E2H is 0, this field behaves as 1 for all purposes other than a direct read of the value of this bit.
- When FEAT_VHE is implemented and HCR_EL2.E2H is 1, this field behaves as 0 for all purposes other

than a direct read of the value of this bit.

For more information, see ‘Asynchronous exception routing’.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FMO, bit [3]

Physical FIQ Routing.

Value	Meaning
`0b0`	When executing at Exception levels below EL2, and EL2 is enabled in the current Security state: • When the value of HCR_EL2.TGE is 0, Physical FIQ interrupts are not taken to EL2. • When the value of HCR_EL2.TGE is 1, Physical FIQ interrupts are taken to EL2 unless they are routed to EL3. • Virtual FIQ interrupts are disabled.
`0b1`	When executing at any Exception level, and EL2 is enabled in the current Security state: • Physical FIQ interrupts are taken to EL2, unless they are routed to EL3. • When HCR_EL2.TGE is 0, then Virtual FIQ interrupts are enabled.

If EL2 is enabled in the current Security state and the value of HCR_EL2.TGE is 1:

- Regardless of the value of the FMO bit, physical FIQ Interrupts target EL2 unless they are routed to EL3.
- When FEAT_VHE is not implemented, or if HCR_EL2.E2H is 0, this field behaves as 1 for all purposes other than a direct read of the value of this bit.
- When FEAT_VHE is implemented and HCR_EL2.E2H is 1, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

For more information, see ‘Asynchronous exception routing’.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

PTW, bit [2]

Protected Table Walk. In the EL1&0 translation regime, a translation table access made as part of a stage 1 translation table walk is subject to a stage 2 translation. The combining of the memory type attributes from the two stages of translation means the access might be made to a type of Device memory. If this occurs, then the value of this bit determines the behavior:

Value	Meaning
`0b0`	The translation table walk occurs as if it is to Normal Non-cacheable memory. This means it can be made speculatively.

Value	Meaning
`0b1`	The memory access generates a stage 2 Permission fault.

This bit is permitted to be cached in a TLB.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SWIO, bit [1]

Set/Way Invalidation Override. Causes EL1 execution of the data cache invalidate by set/way instructions to perform a data cache clean and invalidate by set/way:

Value	Meaning
`0b0`	This control has no effect on the operation of data cache invalidate by set/way instructions.
`0b1`	Data cache invalidate by set/way instructions perform a data cache clean and invalidate by set/way.

When the value of this bit is 1:

AArch32: DCISW performs the same invalidation as a DCCISW instruction.

AArch64: DC ISW performs the same invalidation as a DC CISW instruction.

This bit can be implemented as RES1.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VM, bit [0]

Virtualization enable. Enables stage 2 address translation for the EL1&0 translation regime, when EL2 is enabled in the current Security state.

Value	Meaning
`0b0`	EL1&0 stage 2 address translation disabled.
`0b1`	EL1&0 stage 2 address translation enabled.

When the value of this bit is 1, data cache invalidate instructions executed at EL1 perform a data cache clean and invalidate. For the invalidate by set/way instruction this behavior applies regardless of the value of the HCR_EL2.SWIO bit.

This bit is permitted to be cached in a TLB.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing the HCR_EL2

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, HCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x078];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return HCR_EL2;
elsif PSTATE.EL == EL3 then
    return HCR_EL2;
```

MSR HCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x078] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HCR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    HCR_EL2 = X[t];
```

A2.1.11 HPFAR_EL2, Hypervisor IPA Fault Address Register

The HPFAR_EL2 characteristics are:

Purpose

Holds the faulting IPA for some aborts on a stage 2 translation taken to EL2.

Attributes

HPFAR_EL2 is a 64-bit register.

Configuration

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

The HPFAR_EL2 is written for:

- Translation or Access faults in the second stage of translation.
- An abort in the second stage of translation performed during the translation table walk of a first stage translation, caused by a Translation fault, an Access flag fault, or a Permission fault.
- A stage 2 Address size fault.
- If FEAT_RME is implemented, a Granule Protection Check fault in the second stage of translation.

For all other exceptions taken to EL2, this register is UNKNOWN.

The address held in this register is an address accessed by the instruction fetch or data access that caused the exception that gave rise to the Instruction Abort or Data Abort. It is the lowest address that gave rise to the fault. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores an unaligned address that crosses a page boundary, the architecture does not prioritize between those different faults.

AArch64 system register HPFAR_EL2 bits [31:0] are architecturally mapped to AArch32 system register HPFAR[31:0].

Field descriptions

The HPFAR_EL2 bit assignments are:

63	62–44	43–32
NS	RES0	FIPA

31–4	3–0
FIPA	RES0

Execution at EL1 or EL0 makes HPFAR_EL2 become UNKNOWN.

NS, bit [63]

When FEAT_SEL2 is implemented:

Faulting IPA address space.

Value	Meaning
`0b0`	Faulting IPA is from the Secure IPA space.
`0b1`	Faulting IPA is from the Non-secure IPA space.

For Data Aborts or Instruction Aborts taken to Non-secure EL2:

- This field is RES0.
- The address is from the Non-secure IPA space.

If FEAT_RME is implemented, for Data Aborts or Instruction Aborts taken to Realm EL2:

- This field is RES0.
- The address is from the Realm IPA space.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

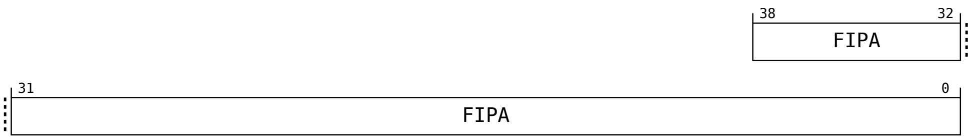
RES0

Bits [62:44]

Reserved, RES0.

FIPA, bits [43:4]

When FEAT_LPA is implemented:



FIPA, bits [38:0]

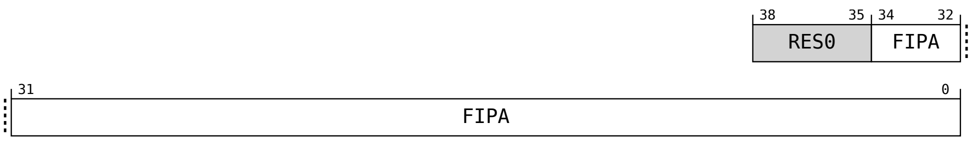
Bits [51:12] of the Faulting Intermediate Physical Address.

For implementations with fewer than 52 physical address bits, the corresponding upper bits in this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_LPA is not implemented:



Bits [38:35]

Reserved, RES0.

FIPA, bits [34:0]

Bits[47:12] Faulting Intermediate Physical Address.

For implementations with fewer than 48 physical address bits, the corresponding upper bits in this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [3:0]

Reserved, RES0.

Accessing the HPFAR_EL2

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, HPFAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return HPFAR_EL2;
elsif PSTATE.EL == EL3 then
    return HPFAR_EL2;
```

MSR HPFAR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HPFAR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    HPFAR_EL2 = X[t];
```

A2.1.12 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0

The ID_AA64PFR0_EL1 characteristics are:

Purpose

Provides additional information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see ‘Principles of the ID scheme for fields in ID registers’.

Attributes

ID_AA64PFR0_EL1 is a 64-bit register.

Configuration

The external register EDPFR gives information from this register.

Field descriptions

The ID_AA64PFR0_EL1 bit assignments are:

63:60	59:56	55:52	51:48	47:44	43:40	39:36	35:32
CSV3	CSV2	RME	DIT	AMU	MPAM	SEL2	SVE

31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
RAS	GIC	AdvSIMD	FP	EL3	EL2	EL1	EL0

CSV3, bits [63:60]

Speculative use of faulting data. Defined values are:

Value	Meaning
0b0000	This PE does not disclose whether data loaded under speculation with a permission or domain fault can be used to form an address or generate condition codes or SVE predicate values to be used by other instructions in the speculative sequence.
0b0001	Data loaded under speculation with a permission or domain fault cannot be used to form an address, generate condition codes, or generate SVE predicate values to be used by other instructions in the speculative sequence. The execution timing of any other instructions in the speculative sequence is not a function of the data loaded under speculation.

All other values are reserved.

FEAT_CSV3 implements the functionality identified by the value 0b0001.

In Armv8.0, the permitted values are 0b0000 and 0b0001.

From Armv8.5, the only permitted value is 0b0001.

If FEAT_E0PD is implemented, FEAT_CSV3 must be implemented.

CSV2, bits [59:56]

Speculative use of out of context branch targets. Defined values are:

Value	Meaning
0b0000	This PE does not disclose whether branch targets trained in one hardware-described context can exploitatively control speculative execution in a different hardware-described context.
0b0001	Branch targets trained in one hardware-described context can exploitatively control speculative execution in a different hardware-described context only in a hard-to-determine way. Contexts do not include the SCXTNUM_ELx register contexts. Support for the SCXTNUM_ELx registers is defined in ID_AA64PFR1_EL1.CSV2_frac.
0b0010	Branch targets trained in one hardware-described context can exploitatively control speculative execution in a different hardware-described context only in a hard-to-determine way. The SCXTNUM_ELx registers are supported and the contexts include the SCXTNUM_ELx register contexts.

All other values are reserved.

FEAT_CSV2 implements the functionality identified by the value 0b0001.

FEAT_CSV2_2 implements the functionality identified by the value 0b0010.

In Armv8.0, the permitted values are 0b0000, 0b0001, and 0b0010.

From Armv8.5, the permitted values are 0b0001 and 0b0010.

RME, bits [55:52]

Realm Management Extension (RME). Defined values are:

Value	Meaning
0b0000	Realm Management Extension not implemented.
0b0001	RMEv1 is implemented.

All other values are reserved.

FEAT_RME implements the functionality identified by the value 0b0001.

DIT, bits [51:48]

Data Independent Timing. Defined values are:

Value	Meaning
0b0000	AArch64 does not guarantee constant execution time of any instructions.

Value	Meaning
0b0001	AArch64 provides the PSTATE.DIT mechanism to guarantee constant execution time of certain instructions.

All other values are reserved.

FEAT_DIT implements the functionality identified by the value 0b0001.

From Armv8.4, the only permitted value is 0b0001.

AMU, bits [47:44]

Indicates support for Activity Monitors Extension. Defined values are:

Value	Meaning
0b0000	Activity Monitors Extension is not implemented.
0b0001	FEAT_AMUv1 is implemented.
0b0010	FEAT_AMUv1p1 is implemented. As 0b0001 and adds support for virtualization of the activity monitor event counters.

All other values are reserved.

FEAT_AMUv1 implements the functionality identified by the value 0b0001.

FEAT_AMUv1p1 implements the functionality identified by the value 0b0010.

In Armv8.0, the only permitted value is 0b0000.

In Armv8.4, the permitted values are 0b0000 and 0b0001.

From Armv8.6, the permitted values are 0b0000, 0b0001, and 0b0010.

MPAM, bits [43:40]

Indicates support for MPAM Extension. Defined values are:

Value	Meaning
0b0000	If ID_AA64PFR1_EL1.MPAM_frac == 0b0000, MPAM Extension is not implemented. If ID_AA64PFR1_EL1.MPAM_frac == 0b0001, MPAM Extension version 0.1 is implemented.
0b0001	If ID_AA64PFR1_EL1.MPAM_frac == 0b0000, MPAM Extension version 1.0 is implemented. If ID_AA64PFR1_EL1.MPAM_frac == 0b0001, MPAM Extension version 1.1 is implemented.

All other values are reserved.

SEL2, bits [39:36]

Secure EL2. Defined values are:

Value	Meaning
0b0000	Secure EL2 is not implemented.
0b0001	Secure EL2 is implemented.

All other values are reserved.

FEAT_SEL2 implements the functionality identified by the value 0b0001.

SVE, bits [35:32]

Scalable Vector Extension. Defined values are:

Value	Meaning
0b0000	SVE architectural state and programmers' model are not implemented.
0b0001	SVE architectural state and programmers' model are implemented.

All other values are reserved.

If implemented, refer to ID_AA64ZFR0_EL1 for information about which SVE instructions are available.

RAS, bits [31:28]

RAS Extension version. Defined values are:

Value	Meaning
0b0000	No RAS Extension.
0b0001	RAS Extension implemented.
0b0010	FEAT_RASv1p1 implemented and, if EL3 is implemented, FEAT_DoubleFault implemented. As 0b0001, and adds support for: • If EL3 is implemented, FEAT_DoubleFault. • Additional ERXMISC<m>_EL1 System registers. • Additional System registers ERXPFGCDN_EL1, ERXPFGCTL_EL1, and ERXPFGF_EL1, and the SCR_EL3.FIEN and HCR_EL2.FIEN trap controls, to support the optional RAS Common Fault Injection Model Extension. Error records accessed through System registers conform to RAS System Architecture v1.1, which includes simplifications to ERR<n>STATUS and support for the optional RAS Timestamp and RAS Common Fault Injection Model Extensions.

All other values are reserved.

FEAT_RAS implements the functionality identified by the value 0b0001.

FEAT_RASv1p1 and FEAT_DoubleFault implement the functionality identified by the value 0b0010.

In Armv8.0 and Armv8.1, the permitted values are 0b0000 and 0b0001.

In Armv8.2, the only permitted value is 0b0001.

From Armv8.4, if FEAT_DoubleFault is implemented, the only permitted value is 0b0010.

From Armv8.4, when FEAT_DoubleFault is not implemented, and ERRIDR_EL1 is 0, the permitted values are IMPLEMENTATION DEFINED 0b0001 or 0b0010.

When the value of this field is 0b0001, ID_AA64PFR1_EL1.RAS_frac indicates whether FEAT_RASv1p1 is implemented.

GIC, bits [27:24]

System register GIC CPU interface. Defined values are:

Value	Meaning
0b0000	GIC CPU interface system registers not implemented.
0b0001	System register interface to versions 3.0 and 4.0 of the GIC CPU interface is supported.
0b0011	System register interface to version 4.1 of the GIC CPU interface is supported.

All other values are reserved.

AdvSIMD, bits [23:20]

Advanced SIMD. Defined values are:

Value	Meaning
0b0000	Advanced SIMD is implemented, including support for the following SISD and SIMD operations: • Integer byte, halfword, word and doubleword element operations. • Single-precision and double-precision floating-point arithmetic. • Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.
0b0001	As for 0b0000, and also includes support for half-precision floating-point arithmetic.
0b1111	Advanced SIMD is not implemented.

All other values are reserved.

This field must have the same value as the FP field.

The permitted values are:

- 0b0000 in an implementation with Advanced SIMD support that does not include the FEAT_FP16 extension.
- 0b0001 in an implementation with Advanced SIMD support that includes the FEAT_FP16 extension.
- 0b1111 in an implementation without Advanced SIMD support.

FP, bits [19:16]

Floating-point. Defined values are:

Value	Meaning
0b0000	Floating-point is implemented, and includes support for: • Single-precision and double-precision floating-point types. • Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.
0b0001	As for 0b0000, and also includes support for half-precision floating-point arithmetic.
0b1111	Floating-point is not implemented.

All other values are reserved.

This field must have the same value as the AdvSIMD field.

The permitted values are:

- 0b0000 in an implementation with floating-point support that does not include the FEAT_FP16 extension.
- 0b0001 in an implementation with floating-point support that includes the FEAT_FP16 extension.
- 0b1111 in an implementation without floating-point support.

EL3, bits [15:12]

EL3 Exception level handling. Defined values are:

Value	Meaning
0b0000	EL3 is not implemented.
0b0001	EL3 can be executed in AArch64 state only.
0b0010	EL3 can be executed in either AArch64 or AArch32 state.

All other values are reserved.

EL2, bits [11:8]

EL2 Exception level handling. Defined values are:

Value	Meaning
0b0000	EL2 is not implemented.

Value	Meaning
0b0001	EL2 can be executed in AArch64 state only.
0b0010	EL2 can be executed in either AArch64 or AArch32 state.

All other values are reserved.

EL1, bits [7:4]

EL1 Exception level handling. Defined values are:

Value	Meaning
0b0001	EL1 can be executed in AArch64 state only.
0b0010	EL1 can be executed in either AArch64 or AArch32 state.

All other values are reserved.

EL0, bits [3:0]

EL0 Exception level handling. Defined values are:

Value	Meaning
0b0001	EL0 can be executed in AArch64 state only.
0b0010	EL0 can be executed in either AArch64 or AArch32 state.

All other values are reserved.

Accessing the ID_AA64PFR0_EL1

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, ID_AA64PFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b000

```
if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL2 then
```

```
    return ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64PFR0_EL1;
```

A2.1.13 MDCR_EL3, Monitor Debug Configuration Register (EL3)

The MDCR_EL3 characteristics are:

Purpose

Provides EL3 configuration options for self-hosted debug and the Performance Monitors Extension.

Attributes

MDCR_EL3 is a 64-bit register.

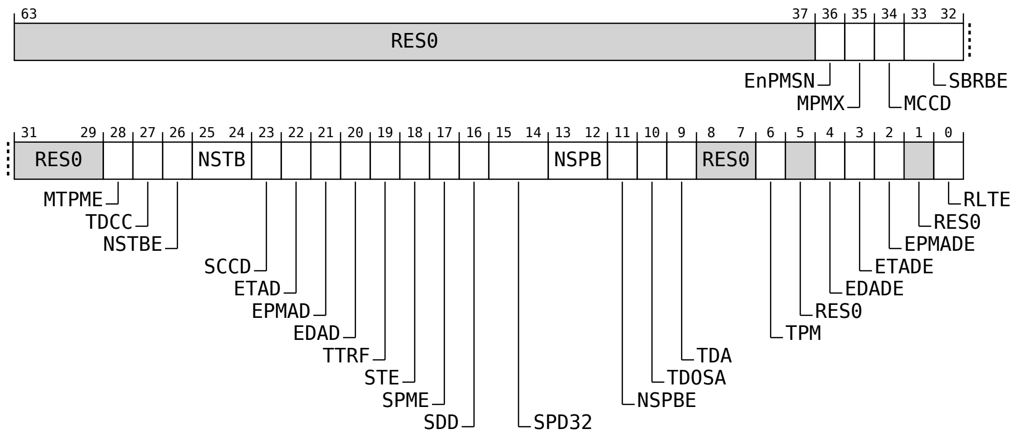
Configuration

AArch64 system register MDCR_EL3 bits [31:0] can be mapped to AArch32 system register SDCR[31:0], but this is not architecturally mandated.

This register is present only when EL3 is implemented. Otherwise, direct accesses to MDCR_EL3 are UNDEFINED.

Field descriptions

The MDCR_EL3 bit assignments are:



Bits [63:37]

Reserved, RES0.

EnPMSN, bit [36]

When FEAT_SPEv1p2 is implemented:

Trap accesses to PMSNEVFR_EL1. Controls access to Statistical Profiling PMSNEVFR_EL1 System register from EL2 and EL1.

Value	Meaning
`0b0`	Accesses to PMSNEVFR_EL1 at EL2 and EL1 generate a Trap exception to EL3.
`0b1`	Do not trap PMSNEVFR_EL1 to EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

MPMX, bit [35]

When FEAT_PMUv3p7 is implemented:

Monitor Performance Monitors Extended control. In conjunction with MDCR_EL3.SPME, controls when event counters are enabled at EL3 and in other Secure Exception levels.

Value	Meaning
`0b0`	Event counting and PMCCNTR_EL0 are not affected by this mechanism.
`0b1`	Event counting by some or all event counters is prohibited at EL3. If PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled at EL3. Otherwise, PMCCNTR_EL0 is not affected by this mechanism.

If EL2 is implemented, MDCR_EL3.SPME == 1, and MDCR_EL2.HPMN is less than PMCR_EL0.N then all the following are true:

- If MDCR_EL2.HPMN is not 0, this field affects the operation of event counters in the range [0 .. (MDCR_EL2.HPMN-1)] at EL3.
- This field does not affect the operation of other event counters.
- If PMCR_EL0.DP is 1, this field affects the operation of PMCCNTR_EL0 at EL3.

The operation of this field applies even when EL2 is disabled in the current Security state.

If EL2 is not implemented, MDCR_EL3.SPME == 0, or MDCR_EL2.HPMN is equal to PMCR_EL0.N then this field affects the operation of all event counters at EL3, and if PMCR_EL0.DP is 1, the operation of PMCCNTR_EL0 at EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

MCCD, bit [34]

When FEAT_PMUv3p7 is implemented:

Monitor Cycle Counter Disable. Prohibits the Cycle Counter, PMCCNTR_EL0, from counting at EL3.

Value	Meaning
`0b0`	Cycle counting by PMCCNTR_EL0 is not affected by this mechanism.
`0b1`	Cycle counting by PMCCNTR_EL0 is prohibited at EL3.

This field does not affect the CPU_CYCLES event or any other event that counts cycles.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

SBRBE, bits [33:32]

When FEAT_BRBE is implemented:

Secure Branch Record Buffer Enable. Controls branch recording by the BRBE, and access to BRBE registers and instructions at EL2 and EL1.

Value	Meaning
`0b00`	Direct accesses to BRBE registers and instructions, except when in EL3, generate a Trap exception to EL3. EL0, EL1, and EL2 are prohibited regions.
`0b01`	Direct accesses to BRBE registers and instructions in Secure state, except when in EL3, generate a Trap exception to EL3. EL0, EL1, and EL2 in Secure state are prohibited regions. This control does not cause any direct accesses to BRBE registers when not in Secure state to be trapped, and does not cause any Exception levels when not in Secure state to be a prohibited region.
`0b10`	Direct accesses to BRBE registers and instructions, except when in EL3, generate a Trap exception to EL3. This control does not cause any Exception levels to be prohibited regions.
`0b11`	This control does not cause any direct accesses to BRBE registers or instruction to be trapped, and does not cause any Exception levels to be a prohibited region.

The Branch Record Buffer registers trapped by this control are: BRBCR_EL1, BRBCR_EL2, BRBCR_EL12, BRBFCR_EL1, BRBIDR0_EL1, BRBINF<n>_EL1, BRBINFINJ_EL1, BRBSRC<n>_EL1, BRBSRCINJ_EL1, BRBTGT<n>_EL1, BRBTGTINJ_EL1, and BRBTS_EL1.

The Branch Record Buffer instructions trapped by this control are:

- BRB IALL.
- BRB INJ.

EL3 is a prohibited region.

If EL3 is not implemented then the Effective value of this field is 0b11.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [31:29]

Reserved, RES0.

MTPME, bit [28]

When FEAT_MTPMU is implemented:

Multi-threaded PMU Enable. Enables use of the PMEVTYPER<n>_EL0.MT bits.

Value	Meaning
0b0	FEAT_MTPMU is disabled. The Effective value of PMEVTYPER<n>_EL0.MT is zero.
0b1	PMEVTYPER<n>_EL0.MT bits not affected by this field.

If FEAT_MTPMU is disabled for any other PE in the system that has the same level 1 Affinity as the PE, it is IMPLEMENTATION DEFINED whether the PE behaves as if this field is 0.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0b1.

Otherwise:

RES0

TDCC, bit [27]

When FEAT_FGT is implemented:

Trap DCC. Traps use of the Debug Comms Channel at EL2, EL1, and EL0 to EL3.

Value	Meaning
0b0	This control does not cause any register accesses to be trapped.
0b1	Accesses to the DCC registers at EL2, EL1, and EL0 generate a Trap exception to EL3, unless the access also generates a higher priority exception. Traps on the DCC data transfer registers are ignored when the PE is in Debug state.

The DCC registers trapped by this control are:

AArch64: OSDTRRX_EL1, OSDTRTX_EL1, MDCCSR_EL0, MDCCINT_EL1, and, when the PE is in Non-debug state, DBGDTR_EL0, DBGDTRRX_EL0, and DBGDTRTX_EL0.

AArch32: DBGDTRRXext, DBGDTRTXext, DBGDSCRint, DBGDCCINT, and, when the PE is in Non-debug state, DBGDTRRXint and DBGDTRTXint.

The traps are reported with EC syndrome value:

- 0x05 for trapped AArch32 MRC and MCR accesses with coproc == 0b1110.
- 0x06 for trapped AArch32 LDC to DBGDTRTXint and STC from DBGDTRRXint.
- 0x18 for trapped AArch64 MRS and MSR accesses.

When the PE is in Debug state, MDCR_EL3.TDCC does not trap any accesses to:

AArch64: DBGDTR_EL0, DBGDTRRX_EL0, and DBGDTRTX_EL0.

AArch32: DBGDTRRXint and DBGDTRTXint.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NSTBE, bit [26]

When FEAT_TRBE is implemented and FEAT_RME is implemented:

Non-secure Trace Buffer Extended. Together with MDCR_EL3.NSTB, controls the owning translation regime and accesses to Trace Buffer control registers from EL2 and EL1.

For a description of the values derived by evaluating NSTB and NSTBE together, see MDCR_EL3.NSTB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [25:24]

When FEAT_TRBE is implemented and FEAT_RME is implemented

NSTB, bits [25:24]

Non-secure Trace Buffer. Together with MDCR_EL3.NSTBE, controls the owning translation regime and accesses to Trace Buffer control registers from EL2 and EL1.

Value	Meaning
`0b00`	When MDCR_EL3.NSTBE == 0b0: Trace Buffer owning security state is Secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3. When MDCR_EL3.NSTBE == 0b1: Reserved.
`0b01`	When MDCR_EL3.NSTBE == 0b0: Trace Buffer owning security state is Secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 in Non-secure and Realm state generate Trap exceptions to EL3. When MDCR_EL3.NSTBE == 0b1: Reserved.

Value	Meaning
`0b10`	When MDCR_EL3.NSTBE == 0b0: Trace Buffer owning security state is Non-secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3. When MDCR_EL3.NSTBE == 0b1: Trace Buffer owning security state is Realm state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Secure state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
`0b11`	When MDCR_EL3.NSTBE == 0b0: Trace Buffer owning security state is Non-secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 in Secure and Realm state generate Trap exceptions to EL3. When MDCR_EL3.NSTBE == 0b1: Trace Buffer owning security state is Realm state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Secure state. Accesses to Trace Buffer control registers at EL2 and EL1 in Non-secure and Secure state generate Trap exceptions to EL3.

The Trace Buffer control registers trapped by this control are: TRBBASER_EL1, TRBLIMITR_EL1, TRBMAR_EL1, TRBPTR_EL1, TRBSR_EL1, and TRBTRG_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_TRBE is implemented and FEAT_RME is not implemented

NSTB, bits [25:24]

Non-secure Trace Buffer. Controls the owning translation regime and accesses to Trace Buffer control registers from EL2 and EL1.

Value	Meaning
`0b00`	Trace Buffer owning security state is Secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
`0b01`	Trace Buffer owning security state is Secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure state. Accesses to Trace Buffer control registers at EL2 and EL1 in Non-secure state generate Trap exceptions to EL3.

Value	Meaning
0b10	Trace Buffer owning security state is Non-secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Secure state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
0b11	Trace Buffer owning security state is Non-secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Secure state. Accesses to Trace Buffer control registers at EL2 and EL1 in Secure state generate Trap exceptions to EL3.

The Trace Buffer control registers trapped by this control are: TRBBASER_EL1, TRBLIMITR_EL1, TRBMAR_EL1, TRBPTR_EL1, TRBSR_EL1, and TRBTRG_EL1.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 1, then the Effective value of this field is 0b11.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0, then the Effective value of this field is 0b01.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SCCD, bit [23]

When FEAT_PMUv3p5 is implemented:

Secure Cycle Counter Disable. Prohibits PMCCNTR_EL0 from counting in Secure state.

Value	Meaning
0b0	Cycle counting by PMCCNTR_EL0 is not affected by this mechanism.
0b1	Cycle counting by PMCCNTR_EL0 is prohibited in Secure state.

This field does not affect the CPU_CYCLES event or any other event that counts cycles.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0b0.

Otherwise:

RES0

Bit [22]

When FEAT_RME is implemented, external debugger access to the PE Trace Unit registers is implemented

and FEAT_TRBE is implemented

ETAD, bit [22]

External Trace Access Disable. Together with MDCR_EL3.ETADE, controls access to PE Trace Unit registers by an external debugger.

ETADE	ETAD	Meaning
0b0	0b0	Access to PE Trace Unit registers by an external debugger is permitted.
0b0	0b1	Root and Secure access to PE Trace Unit registers by an external debugger is permitted. Realm and Non-secure access to PE Trace Unit registers by an external debugger is not permitted.
0b1	0b0	Root and Realm access to PE Trace Unit registers by an external debugger is permitted. Secure and Non-secure access to PE Trace Unit registers by an external debugger is not permitted.
0b1	0b1	Root access to PE Trace Unit registers by an external debugger is permitted. Secure, Non-secure, and Realm access to PE Trace Unit registers by an external debugger is not permitted.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

When external debugger access to the PE Trace Unit registers is implemented and FEAT_TRBE is implemented

ETAD, bit [22]

External Trace Access Disable. Controls Non-secure access to PE Trace Unit registers by an external debugger.

Value	Meaning
`0b0`	Non-secure accesses from an external debugger to PE Trace Unit are allowed.
`0b1`	Non-secure accesses from an external debugger to some PE Trace Unit registers are prohibited. See individual registers for the effect of this field.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0, then the Effective value of this field is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

Bit [21]

When FEAT_RME is implemented, FEAT_PMUv3 is implemented and the Performance Monitors Extension supports external debug interface accesses

EPMAD, bit [21]

External Performance Monitors Access Disable. Together with MDCR_EL3.EPMADE, controls access to Performance Monitor registers by an external debugger.

EPMADE	EPMAD	Meaning
0b0	0b0	Access to Performance Monitor registers by an external debugger is permitted.
0b0	0b1	Root and Secure access to Performance Monitor registers by an external debugger is permitted. Realm and Non-secure access to Performance Monitor registers by an external debugger is not permitted.
0b1	0b0	Root and Realm access to Performance Monitor registers by an external debugger is permitted. Secure and Non-secure access to Performance Monitor registers by an external debugger is not permitted.
0b1	0b1	Root access to Performance Monitor registers by an external debugger is permitted. Secure, Non-secure, and Realm access to Performance Monitor registers by an external debugger is not permitted.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

When FEAT_Debugv8p4 is implemented, FEAT_PMUv3 is implemented and the Performance Monitors Extension supports external debug interface accesses

EPMAD, bit [21]

External Performance Monitors Non-secure Access Disable. Controls Non-secure access to Performance Monitor registers by an external debugger.

Value	Meaning
`0b0`	Non-secure access to Performance Monitor registers from external debugger is permitted.
`0b1`	Non-secure access to Performance Monitor registers from external debugger is not permitted.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0b0, then the Effective value of this bit is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

When FEAT_PMUv3 is implemented and the Performance Monitors Extension supports external debug interface accesses

EPMAD, bit [21]

External Performance Monitors Access Disable. Controls access to Performance Monitor registers by an external debugger.

Value	Meaning
`0b0`	Access to Performance Monitor registers from external debugger is permitted.
`0b1`	Access to Performance Monitor registers from external debugger is not permitted, unless overridden by the IMPLEMENTATION DEFINED authentication interface.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0b0, then the Effective value of this bit is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

Bit [20]

When FEAT_RME is implemented

EDAD, bit [20]

External Debug Access Disable. Together with MDCR_EL3.EDADE, controls access to breakpoint registers, watchpoint registers, and OSLAR_EL1 by an external debugger.

EDADE	EDAD	Meaning
0b0	0b0	Access to Debug registers by an external debugger is permitted.
0b0	0b1	Root and Secure access to Debug registers by an external debugger is permitted. Realm and Non-secure access to Debug registers by an external debugger is not permitted.
0b1	0b0	Root and Realm access to Debug registers by an external debugger is permitted. Secure and Non-secure access to Debug registers by an external debugger is not permitted.
0b1	0b1	Root access to Debug registers by an external debugger is permitted. Secure, Non-secure, and Realm access to Debug registers by an external debugger is not permitted.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

When FEAT_Debugv8p4 is implemented

EDAD, bit [20]

External Debug Non-secure Access Disable. Controls Non-secure access to breakpoint, watchpoint, and OSLAR_EL1 registers by an external debugger.

Value	**Meaning**
`0b0`	Non-secure access to debug registers from external debugger is permitted.
`0b1`	Non-secure access to breakpoint and watchpoint registers, and OSLAR_EL1 from external debugger is not permitted.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0b0, then the Effective value of this field is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

When FEAT_Debugv8p2 is implemented

EDAD, bit [20]

External Debug Access Disable. Controls access to breakpoint, watchpoint, and OSLAR_EL1 registers by an external debugger.

Value	Meaning
`0b0`	Access to debug registers, and to OSLAR_EL1 from external debugger is permitted.
`0b1`	Access to breakpoint and watchpoint registers, and to OSLAR_EL1 from external debugger is not permitted, unless overridden by the IMPLEMENTATION DEFINED authentication interface.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0b0, then the Effective value of this field is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise

EDAD, bit [20]

External Debug Access disable. Controls access to breakpoint, watchpoint, and optionally OSLAR_EL1 registers by an external debugger.

Value	Meaning
`0b0`	Access to debug registers from external debugger is permitted.
`0b1`	Access to breakpoint and watchpoint registers from an external debugger is not permitted, unless overridden by the IMPLEMENTATION DEFINED authentication interface. It is IMPLEMENTATION DEFINED whether access to the OSLAR_EL1 register from an external debugger is permitted or not permitted.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0b0, then the Effective value of this field is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

TTRF, bit [19]

When FEAT_TRF is implemented:

Trap Trace Filter controls. Traps use of the Trace Filter control registers at EL2 and EL1 to EL3.

The Trace Filter registers trapped by this control are:

- TRFCR_EL2, TRFCR_EL12, TRFCR_EL1, reported using EC syndrome value 0x18.

- HTRFCR and TRFCR, reported using EC syndrome value 0x03.

Value	Meaning
0b0	Accesses to Trace Filter registers at EL2 and EL1 are not affected by this bit.
0b1	Accesses to Trace Filter registers at EL2 and EL1 generate a Trap exception to EL3, unless the access generates a higher priority exception.

Otherwise:

RES0

STE, bit [18]

When FEAT_TRF is implemented:

Secure Trace enable. Enables tracing in Secure state.

Value	Meaning
0b0	Trace prohibited in Secure state unless overridden by the IMPLEMENTATION DEFINED authentication interface.
0b1	Trace in Secure state is not affected by this bit.

This bit also controls the level of authentication required by an external debugger to enable external tracing. See 'Register controls to enable self-hosted trace'.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0b0, the Effective value of this bit is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0b0.

Otherwise:

RES0

Bit [17]

When FEAT_PMUv3 is implemented and FEAT_PMUv3p7 is implemented

SPME, bit [17]

Secure Performance Monitors Enable. Controls event counting in Secure state and EL3.

Value	Meaning
0b0	When MDCR_EL3.MPMX == 0: Event counting is prohibited in Secure state. If PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled in Secure state. Otherwise, PMCCNTR_EL0 is not affected by this mechanism.
0b1	When MDCR_EL3.MPMX == 0: Event counting and PMCCNTR_EL0 are not affected by this mechanism.

When MDCR_EL3.MPMX is 0, this field affects the operation of all event counters in Secure state, and if PMCR_EL0.DP is 1, the operation of PMCCNTR_EL0 in Secure state.

When MDCR_EL3.MPMX is 1, this field affects the operation of event counters at EL3 only, and if PMCR_EL0.DP is 1, the operation of PMCCNTR_EL0 at EL3 only. See MDCR_EL3.MPMX for more information.

When PMCR_EL0.DP is 0, PMCCNTR_EL0 is not affected by this field.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0, then the Effective value of this field is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

When FEAT_PMUv3 is implemented and FEAT_Debugv8p2 is implemented

SPME, bit [17]

Secure Performance Monitors Enable. Controls event counting in Secure state.

Value	Meaning
`0b0`	Event counting is prohibited in Secure state. If PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled in Secure state. Otherwise, PMCCNTR_EL0 is not affected by this mechanism.
`0b1`	Event counting and PMCCNTR_EL0 are not affected by this mechanism.

This field affects the operation of all event counters in Secure state, and if PMCR_EL0.DP is 1, the operation of PMCCNTR_EL0 in Secure state. When PMCR_EL0.DP is 0, PMCCNTR_EL0 is not affected by this field.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0, then the Effective value of this field is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

When FEAT_PMUv3 is implemented

SPME, bit [17]

Secure Performance Monitors Enable. Controls event counting in Secure state.

Value	Meaning
`0b0`	If ExternalSecureNoninvasiveDebugEnabled() is FALSE, event counting is prohibited in Secure state, and if PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled in Secure state.
`0b1`	Event counting and PMCCNTR_EL0 are not affected by this mechanism.

If ExternalSecureNoninvasiveDebugEnabled() is TRUE, the event counters and PMCCNTR_EL0 are not affected by this field.

Otherwise, this field affects the operation of all event counters in Secure state, and if PMCR_EL0.DP is 1, the operation of PMCCNTR_EL0 in Secure state. When PMCR_EL0.DP is 0, PMCCNTR_EL0 is not affected by this field.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0, then the Effective value of this field is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

SDD, bit [16]

AArch64 Secure Self-hosted invasive debug disable. Disables Software debug exceptions in Secure state, other than Breakpoint Instruction exceptions.

Value	Meaning
`0b0`	Debug exceptions in Secure state are not affected by this bit.
`0b1`	Debug exceptions, other than Breakpoint Instruction exceptions, are disabled from all Exception levels in Secure state.

The SDD bit is ignored unless both of the following are true:

- The PE is in Secure state.
- The Effective value of SCR_EL3.RW is 0b1.

If Secure EL2 is implemented and enabled, and Secure EL1 is using AArch32, then:

- If debug exceptions from Secure EL1 are enabled, debug exceptions from Secure EL0 are also enabled.
- Otherwise, debug exceptions from Secure EL0 are enabled only if the value of SDER32_EL3.SUIDEN is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SPD32, bits [15:14]

When EL1 is capable of using AArch32:

AArch32 Secure self-hosted privileged debug. Enables or disables debug exceptions from Secure EL1 using AArch32, other than Breakpoint Instruction exceptions.

Value	Meaning
`0b00`	Legacy mode. Debug exceptions from Secure EL1 are enabled by the IMPLEMENTATION DEFINED authentication interface.
`0b10`	Secure privileged debug disabled. Debug exceptions from Secure EL1 are disabled.
`0b11`	Secure privileged debug enabled. Debug exceptions from Secure EL1 are enabled.

Other values are reserved, and have the CONSTRAINED UNPREDICTABLE behavior that they must have the same behavior as 0b00. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

This field has no effect on Breakpoint Instruction exceptions. These are always enabled.

This field is ignored unless both of the following are true:

- The PE is in Secure state.
- The Effective value of SCR_EL3.RW is 0b0.

If Secure EL1 is using AArch32, then:

- If debug exceptions from Secure EL1 are enabled, then debug exceptions from Secure EL0 are also enabled.
- Otherwise, debug exceptions from Secure EL0 are enabled only if the value of SDER32_EL3.SUIDEN is 0b1.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0b0, then the Effective value of this field is 0b11.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [13:12]

When FEAT_SPE is implemented and FEAT_RME is implemented

NSPB, bits [13:12]

Non-secure Profiling Buffer. Together with MDCR_EL3.NSPBE, controls the owning translation regime and accesses to Statistical Profiling and Profiling Buffer control registers.

Value	Meaning
`0b00`	When MDCR_EL3.NSPBE == 0b0: Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in all Security states generate Trap exceptions to EL3. When MDCR_EL3.NSPBE == 0b1: Reserved.
`0b01`	When MDCR_EL3.NSPBE == 0b0: Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure and Realm states generate Trap exceptions to EL3. When MDCR_EL3.NSPBE == 0b1: Reserved.

Value	Meaning
`0b10`	When MDCR_EL3.NSPBE == 0b0: Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in all Security states generate Trap exceptions to EL3. When MDCR_EL3.NSPBE == 0b1: Profiling Buffer uses Realm Virtual Addresses. Statistical Profiling enabled in Realm state and disabled in Non-secure and Secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in all Security states generate Trap exceptions to EL3.
`0b11`	When MDCR_EL3.NSPBE == 0b0: Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Secure and Realm states generate Trap exceptions to EL3. When MDCR_EL3.NSPBE == 0b1: Profiling Buffer uses Realm Virtual Addresses. Statistical Profiling enabled in Realm state and disabled in Non-secure and Secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure and Secure states generate Trap exceptions to EL3.

The Statistical Profiling and Profiling Buffer control registers trapped by this control are:

- PMBLIMITR_EL1, PMBPTR_EL1, PMBSR_EL1, PMSCR_EL1, PMSCR_EL2, PMSCR_EL12, PMSEVFR_EL1, PMSFCR_EL1, PMSICR_EL1, PMSIDR_EL1, PMSIRR_EL1, and PMSLATFR_EL1.
- If FEAT_SPEv1p2 is implemented, PMSNEVFR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_SPE is implemented and FEAT_RME is not implemented

NSPB, bits [13:12]

Non-secure Profiling Buffer. Controls the owning translation regime and accesses to Statistical Profiling and Profiling Buffer control registers.

Value	Meaning
`0b00`	Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure and Secure states generate Trap exceptions to EL3.

Value	Meaning
`0b01`	Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure state generate Trap exceptions to EL3.
`0b10`	Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure and Secure states generate Trap exceptions to EL3.
`0b11`	Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Secure state generate Trap exceptions to EL3.

The Statistical Profiling and Profiling Buffer control registers trapped by this control are:

- PMBLIMITR_EL1, PMBPTR_EL1, PMBSR_EL1, PMSCR_EL1, PMSCR_EL2, PMSCR_EL12, PMSEVFR_EL1, PMSFCR_EL1, PMSICR_EL1, PMSIDR_EL1, PMSIRR_EL1, and PMSLATFR_EL1.
- If FEAT_SPEv1p2 is implemented, PMSNEVFR_EL1.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 1, then the Effective value of this field is 0b11.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0, then the Effective value of this field is 0b01.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NSPBE, bit [11]

When FEAT_RME is implemented:

Non-secure Profiling Buffer Extended. Together with MDCR_EL3.NSPB, controls the owning translation regime and accesses to Statistical Profiling and Profiling Buffer control registers.

For a description of the values derived by evaluating NSPB and NSPBE together, see MDCR_EL3.NSPB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bit [10]

When FEAT_DoubleLock is implemented

TDOSA, bit [10]

Trap debug OS-related register access. Traps EL2 and EL1 System register accesses to the powerdown debug registers to EL3.

Accesses to the registers are trapped as follows:

- Accesses from AArch64 state, OSLAR_EL1, OSLSR_EL1, OSDLR_EL1, DBGPRCR_EL1, and any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit, are trapped to EL3 and reported using EC syndrome value 0x18.
- Accesses using MCR or MRC to DBGOSLAR, DBGOSLSR, DBGOSDLR, and DBGPRCR, are trapped to EL3 and reported using EC syndrome value 0x05.
- Accesses to any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	EL2 and EL1 System register accesses to the powerdown debug registers are trapped to EL3, unless it is trapped by HDCR.TDOSA or MDCR_EL2.TDOSA.

The powerdown debug registers are not accessible at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

TDOSA, bit [10]

Trap debug OS-related register access. Traps EL2 and EL1 System register accesses to the powerdown debug registers to EL3.

The following registers are affected by this trap:

- AArch64: OSLAR_EL1, OSLSR_EL1, and DBGPRCR_EL1.
- AArch32: DBGOSLAR, DBGOSLSR, and DBGPRCR.
- AArch64 and AArch32: Any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.
- It is IMPLEMENTATION DEFINED whether accesses to OSDLR_EL1 and DBGOSDLR are trapped.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	EL2 and EL1 System register accesses to the powerdown debug registers are trapped to EL3, unless it is trapped by HDCR.TDOSA or MDCR_EL2.TDOSA.

The powerdown debug registers are not accessible at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TDA, bit [9]

Trap Debug Access. Traps EL2, EL1, and EL0 System register accesses to those debug System registers that cannot be trapped using the MDCR_EL3.TDOSA field.

Accesses to the debug registers are trapped as follows:

- In AArch64 state, the following registers are trapped to EL3 and reported using EC syndrome value 0x18:
  - DBGBVR<n>_EL1, DBGBCR<n>_EL1, DBGWVR<n>_EL1, DBGWCR<n>_EL1, DBGCLAIMSET_EL1, DBGCLAIMCLR_EL1, DBGAUTHSTATUS_EL1, DBGVCR32_EL2.
  - AArch64: MDCR_EL2, MDRAR_EL1, MDCCSR_EL0, MDCCINT_EL1, MDSCR_EL1, OSDTRRX_EL1, OSDTRTX_EL1, OSECCR_EL1.
- In AArch32 state, SDER is trapped to EL3 and reported using EC syndrome value 0x03.
- In AArch32 state, accesses using MCR or MRC to the following registers are reported using EC syndrome value 0x05, accesses using MCRR or MRRC are reported using EC syndrome value 0x0C:
  - HDCR, DBGDRAR, DBGDSAR, DBGDIDR, DBGDCCINT, DBGWFAR, DBGVCR, DBGBVR<n>, DBGBCR<n>, DBGBXVR<n>, DBGWCR<n>, DBGWVR<n>.
  - DBGCLAIMSET, DBGCLAIMCLR, DBGAUTHSTATUS, DBGDEVID, DBGDEVID1, DBGDEVID2, DBGOSECCR.
- In AArch32 state, STC accesses to DBGDTRRXint and LDC accesses to DBGDTRTXint are reported using EC syndrome value 0x06.
- When not in Debug state, the following registers are also trapped to EL3:
  - AArch64 accesses to DBGDTR_EL0, DBGDTRRX_EL0, and DBGDTRTX_EL0, reported using EC syndrome value 0x18.
  - AArch32 accesses using MCR or MRC to DBGDTRRXint and DBGDTRTXint, reported using EC syndrome value 0x05.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	EL0, EL1, and EL2 accesses to the debug registers, other than the registers that can be trapped by MDCR_EL3.TDOSA, are trapped to EL3, from any Security state and both Execution states, unless it is trapped by DBGDSCRext.UDCCdis, MDSCR_EL1.TDCC, HDCR.TDA or MDCR_EL2.TDA.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [8:7]

Reserved, RES0.

TPM, bit [6]

When FEAT_PMUv3 is implemented:

Trap Performance Monitor register accesses. Accesses to all Performance Monitor registers from EL0, EL1, and EL2 to EL3, from any Security state and both Execution states are trapped as follows:

- In AArch64 state, accesses to the following registers are trapped to EL3 and are reported using EC syndrome value 0x18:
  - PMCR_EL0, PMCNTENSET_EL0, PMCNTENCLR_EL0, PMOVSCLR_EL0, PMSWINC_EL0, PMSELR_EL0, PMCEID0_EL0, PMCEID1_EL0, PMCCNTR_EL0, PMXEVTYPER_EL0, PMXEVCNTR_EL0, PMUSERENR_EL0, PMINTENSET_EL1, PMINTENCLR_EL1, PMOVSSET_EL0, PMEVCNTR<n>_EL0, PMEVTYPER<n>_EL0, PMCCFILTR_EL0.
  - If FEAT_PMUv3p4 is implemented, PMMIR_EL1.
- In AArch32 state, accesses using MCR or MRC to the following registers are reported using EC syndrome value 0x03, accesses using MCRR or MRRC are reported using EC syndrome value 0x04:
  - PMCR, PMCNTENSET, PMCNTENCLR, PMOVSR, PMSWINC, PMSELR, PMCEID0, PMCEID1, PMCCNTR, PMXEVTYPER, PMXEVCNTR, PMUSERENR, PMINTENSET, PMINTENCLR, PMOVSSET, PMEVCNTR<n>, PMEVTYPER<n>, PMCCFILTR.
  - If FEAT_PMUv3p1 is implemented, PMCEID2, and PMCEID3.
  - If FEAT_PMUv3p4 is implemented, PMMIR.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	EL2, EL1, and EL0 System register accesses to all Performance Monitor registers are trapped to EL3, unless it is trapped by HDCR.TPM or MDCR_EL2.TPM.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bit [5]

Reserved, RES0.

EDADE, bit [4]

When FEAT_RME is implemented:

External Debug Access Disable Extended. Together with MDCR_EL3.EDAD, controls access to breakpoint registers, watchpoint registers, and OSLAR_EL1 by an external debugger.

For a description of the values derived by evaluating EDAD and EDADE together, see MDCR_EL3.EDAD.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

ETADE, bit [3]

When FEAT_RME is implemented, external debugger access to the PE Trace Unit registers is implemented and FEAT_TRBE is implemented:

External Trace Access Disable Extended. Together with MDCR_EL3.ETAD, controls access to PE Trace Unit registers by an external debugger.

For a description of the values derived by evaluating ETAD and ETADE together, see MDCR_EL3.ETAD.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

EPMADE, bit [2]

When FEAT_RME is implemented, FEAT_PMUv3 is implemented and the Performance Monitors Extension supports external debug interface accesses:

External Performance Monitors Access Disable Extended. Together with MDCR_EL3.EPMAD, controls access to Performance Monitor registers by an external debugger.

For a description of the values derived by evaluating EPMAD and EPMADE together, see MDCR_EL3.EPMAD.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

Bit [1]

Reserved, RES0.

RLTE, bit [0]

When FEAT_RME is implemented and FEAT_TRF is implemented:

Realm Trace enable. Enables tracing in Realm state.

Value	Meaning
`0b0`	Trace prohibited in Realm state, unless overridden by the IMPLEMENTATION DEFINED authentication interface.
`0b1`	Trace in Realm state is not affected by this bit.

This bit also controls the level of authentication that is required by an external debugger to enable external tracing.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

Accessing the MDCR_EL3

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, MDCR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0011	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return MDCR_EL3;
```

MSR MDCR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0011	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    MDCR_EL3 = X[t];
```

A2.1.14 MFAR_EL3, PA Fault Address Register

The MFAR_EL3 characteristics are:

Purpose

Holds the faulting physical address for Granule Protection Check exceptions taken to EL3.

Attributes

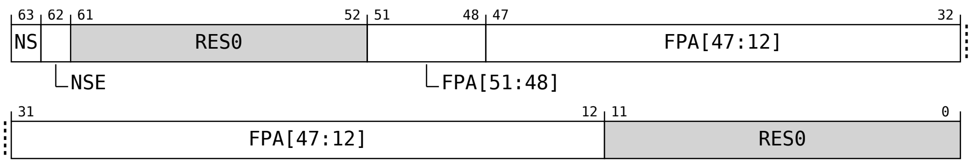
MFAR_EL3 is a 64-bit register.

Configuration

This register is present only when FEAT_RME is implemented. Otherwise, direct accesses to MFAR_EL3 are UNDEFINED.

Field descriptions

The MFAR_EL3 bit assignments are:



An exception return at EL3 makes MFAR_EL3 UNKNOWN.

NS, bit [63]

Together with the NSE field, reports the physical address space of the access that triggered the Granule Protection Check exception.

NSE	NS	Meaning
0b0	0b0	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSE, bit [62]

Together with the NS field, reports the physical address space of the access that triggered the Granule Protection Check exception.

For a description of the values derived by evaluating NS and NSE together, see MFAR_EL3.NS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [61:52]

Reserved, RES0.

FPA[51:48], bits [51:48]

When FEAT_LPA is implemented:

When FEAT_LPA is implemented, extension to FPA[47:12].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

FPA[47:12], bits [47:12]

Bits [47:12] of the faulting physical address.

For implementations with fewer than 48 physical address bits, the corresponding upper bits in this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:0]

Reserved, RES0.

Accessing the MFAR_EL3

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, MFAR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0110	0b0000	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return MFAR_EL3;
```

MSR MFAR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0110	0b0000	0b101

```
if PSTATE.EL == EL0 then
```

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    MFAR_EL3 = X[t];
```

A2.1.15 PAR_EL1, Physical Address Register

The PAR_EL1 characteristics are:

Purpose

Returns the output address (OA) from an Address translation instruction that executed successfully, or fault information if the instruction did not execute successfully.

Attributes

PAR_EL1 is a 64-bit register.

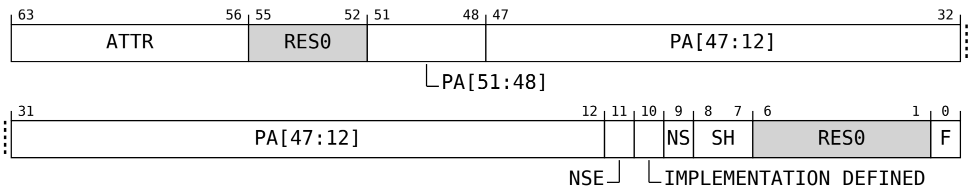
Configuration

AArch64 system register PAR_EL1 bits [63:0] are architecturally mapped to AArch32 system register PAR[63:0].

Field descriptions

The PAR_EL1 bit assignments are:

When PAR_EL1.F == 0:



This section describes the register value returned by the successful execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

On a successful conversion, the PAR_EL1 can return a value that indicates the resulting attributes, rather than the values that appear in the translation table descriptors. More precisely:

- The PAR_EL1.{ATTR, SH} fields are permitted to report the resulting attributes, as determined by any permitted implementation choices and any applicable configuration bits, instead of reporting the values that appear in the translation table descriptors.
- See the PAR_EL1.NS bit description for constraints on the value it returns.

ATTR, bits [63:56]

Memory attributes for the returned output address. This field uses the same encoding as the Attr<n> fields in MAIR_EL1, MAIR_EL2, and MAIR_EL3.

The value returned in this field can be the resulting attribute that is actually implemented by the implementation, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.

The attributes presented are consistent with the stages of translation applied in the address translation instruction. If the instruction performed a stage 1 translation only, the attributes are from the stage 1 translation. If the instruction performed a stage 1 and stage 2 translation, the attributes are from the combined stage 1 and stage 2 translation.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [55:52]

Reserved, RES0.

PA[51:48], bits [51:48]

When FEAT_LPA is implemented:

Extension to PA[47:12]. For more information, see PA[47:12].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

PA[47:12], bits [47:12]

Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[47:12].

When FEAT_LPA is implemented and 52-bit addresses are in use, PA[51:48] forms the upper part of the address value. Otherwise, when 52-bit addresses are not in use, PA[51:48] is RES0.

For implementations with fewer than 48 physical address bits, the corresponding upper bits in this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSE, bit [11]

When FEAT_RME is implemented:

Reports the NSE attribute for a translation table entry from the EL3 translation regime.

For a description of the values derived by evaluating NS and NSE together, see PAR_EL1.NS.

For a result from a Secure, Non-secure, or Realm translation regime, this bit is UNKNOWN.

Otherwise:

RES1

IMPLEMENTATION_DEFINED, bit [10]

IMPLEMENTATION DEFINED

Bit [9]

When FEAT_RME is implemented

NS, bit [9]

Non-secure. The NS attribute for a translation table entry from a Secure translation regime, a Realm translation regime, and the EL3 translation regime.

For a result from an EL3 translation regime, NS and NSE are evaluated together to report the physical address space:

NSE	NS	Meaning
0b0	0b0	Secure.

NSE	NS	Meaning
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

For a result from a Secure translation regime, when SCR_EL3.EEL2 is 1, this bit reflects the Security state of the intermediate physical address space of the translation for the instructions:

- In AArch64 state: AT S1E1R, AT S1E1W, AT S1E1RP, AT S1E1WP, AT S1E0R, and AT S1E0W.
- In AArch32 state: ATS1CPR, ATS1CPW, ATS1CPRP, ATS1CPWP, ATS1CUR, and ATS1CUW.

Otherwise, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.

For a result from a Non-secure translation regime, this bit is UNKNOWN.

For a result from an S1E1 or S1E0 operation on the Realm EL1&0 translation regime, this bit is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

NS, bit [9]

Non-secure. The NS attribute for a translation table entry from a Secure translation regime.

For a result from a Secure translation regime, when SCR_EL3.EEL2 is 1, this bit reflects the Security state of the intermediate physical address space of the translation for the instructions:

- In AArch64 state: AT S1E1R, AT S1E1W, AT S1E1RP, AT S1E1WP, AT S1E0R, and AT S1E0W.
- In AArch32 state: ATS1CPR, ATS1CPW, ATS1CPRP, ATS1CPWP, ATS1CUR, and ATS1CUW.

Otherwise, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.

For a result from a Non-secure translation regime, this bit is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH, bits [8:7]

Shareability attribute, for the returned output address.

Value	Meaning
`0b00`	Non-shareable.
`0b10`	Outer Shareable.
`0b11`	Inner Shareable.

The value 0b01 is reserved.

This field returns the value 0b10 for:

- Any type of Device memory.
- Normal memory with both Inner Non-cacheable and Outer Non-cacheable attributes.

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [6:1]

Reserved, RES0.

F, bit [0]

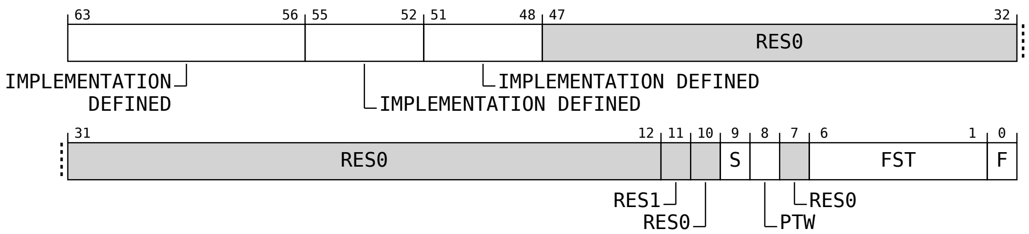
Indicates whether the instruction performed a successful address translation.

Value	Meaning
0b0	Address translation completed successfully.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When PAR_EL1.F == 1:



This section describes the register value returned by a fault on the execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

IMPLEMENTATION_DEFINED, bits [63:56]

IMPLEMENTATION DEFINED

IMPLEMENTATION_DEFINED, bits [55:52]

IMPLEMENTATION DEFINED

IMPLEMENTATION_DEFINED, bits [51:48]

IMPLEMENTATION DEFINED

Bits [47:12]

Reserved, RES0.

Bit [11]

Reserved, RES1.

Bit [10]

Reserved, RES0.

S, bit [9]

Indicates the translation stage at which the translation aborted:

Value	Meaning
`0b0`	Translation aborted because of a fault in the stage 1 translation.
`0b1`	Translation aborted because of a fault in the stage 2 translation.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

PTW, bit [8]

If this bit is set to 1, it indicates the translation aborted because of a stage 2 fault during a stage 1 translation table walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [7]

Reserved, RES0.

FST, bits [6:1]

Fault status code, as shown in the Data Abort ESR encoding.

Value	Meaning	Applies
`0b000000`	Address size fault, level 0 of translation or translation table base register.	
`0b000001`	Address size fault, level 1.	
`0b000010`	Address size fault, level 2.	
`0b000011`	Address size fault, level 3.	
`0b000100`	Translation fault, level 0.	
`0b000101`	Translation fault, level 1.	
`0b000110`	Translation fault, level 2.	
`0b000111`	Translation fault, level 3.	
`0b001001`	Access flag fault, level 1.	
`0b001010`	Access flag fault, level 2.	
`0b001011`	Access flag fault, level 3.	
`0b001000`	Access flag fault, level 0.	When FEAT_LPA2 is implemented
`0b001100`	Permission fault, level 0.	When FEAT_LPA2 is implemented
`0b001101`	Permission fault, level 1.	
`0b001110`	Permission fault, level 2.	
`0b001111`	Permission fault, level 3.	
`0b010011`	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
`0b010100`	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
`0b010101`	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
`0b010110`	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
`0b010111`	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
`0b011011`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
`0b011100`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
`0b011101`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
`0b011110`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented

Value	Meaning	Applies
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
0b111101	Section Domain fault, from an AArch32 stage 1 EL1&0 translation regime using Short-descriptor translation table format.	When EL1 is capable of using AArch32
0b111110	Page Domain fault, from an AArch32 stage 1 EL1&0 translation regime using Short-descriptor translation table format.	When EL1 is capable of using AArch32

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [0]

Indicates whether the instruction performed a successful address translation.

Value	Meaning
0b1	Address translation aborted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing the PAR_EL1

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, PAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0111	0b0100	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGRTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return PAR_EL1;
elsif PSTATE.EL == EL2 then
    return PAR_EL1;
elsif PSTATE.EL == EL3 then
    return PAR_EL1;
```

MSR PAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0111	0b0100	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGWTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PAR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    PAR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    PAR_EL1 = X[t];
```

A2.1.16 PMBIDR_EL1, Profiling Buffer ID Register

The PMBIDR_EL1 characteristics are:

Purpose

Provides information to software as to whether the buffer can be programmed at the current Exception level.

Attributes

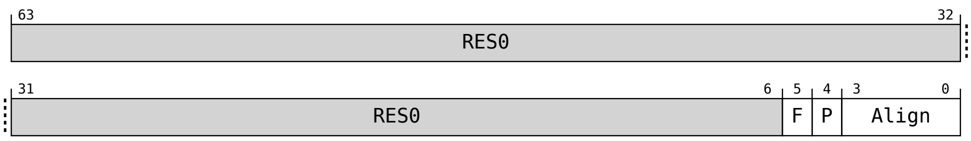
PMBIDR_EL1 is a 64-bit register.

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMBIDR_EL1 are UNDEFINED.

Field descriptions

The PMBIDR_EL1 bit assignments are:



Bits [63:6]

Reserved, RES0.

F, bit [5]

Flag updates. Describes how address translations performed by the Statistical Profiling Extension manage the Access flag and dirty state.

Value	Meaning
0b0	Hardware management of the Access flag and dirty state for accesses made by the Statistical Profiling Extension is always disabled for all translation stages.
0b1	Hardware management of the Access flag and dirty state for accesses made by the Statistical Profiling Extension is controlled in the same way as explicit memory accesses in the Profiling Buffer owning translation regime.

If hardware management of the Access flag is disabled for a stage of translation, an access to a Page or Block with the Access flag bit not set in the descriptor will generate an Access Flag fault.

If hardware management of the dirty state is disabled for a stage of translation, an access to a Page or Block will ignore the Dirty Bit Modifier in the descriptor and might generate a Permission fault, depending on the values of the access permission bits in the descriptor.

Access to this field is **RO**.

P, bit [4]

Programming not allowed. When read at EL3, this field reads as zero. Otherwise, indicates that the Profiling Buffer is owned by a higher Exception level or another Security state. Defined values are:

Value	Meaning
`0b0`	Programming is allowed.
`0b1`	Programming not allowed.

The value read from this field depends on the current Exception level and the Effective values of MDCR_EL3.NSPB, MDCR_EL3.NSPBE, and MDCR_EL2.E2PB:

- If EL3 is implemented, and the owning Security state is Secure state, this field reads as one from:
  - Non-secure EL1 and Non-secure EL2.
  - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
  - If Secure EL2 is implemented and enabled, and MDCR_EL2.E2PB is 0b00, Secure EL1.
- If EL3 is implemented, and the owning Security state is Non-secure state, this field reads as one from:
  - Secure EL1.
  - If Secure EL2 is implemented, Secure EL2.
  - If EL2 is implemented and MDCR_EL2.E2PB is 0b00, Non-secure EL1.
  - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
- If FEAT_RME is implemented, and the owning Security state is Realm state, this field reads as one from:
  - Non-secure EL1 and Non-secure EL2.
  - Secure EL1 and Secure EL2.
  - If MDCR_EL2.E2PB is 0b00, Realm EL1.
- If EL3 is not implemented, EL2 is implemented, and MDCR_EL2.E2PB is 0b00, this field reads as one from EL1.
- Otherwise, this field reads as zero.

Align, bits [3:0]

Defines the minimum alignment constraint for writes to PMBPTR_EL1. Defined values are:

Value	Meaning
`0b0000`	Byte.
`0b0001`	Halfword.
`0b0010`	Word.
`0b0011`	Doubleword.
`0b0100`	16 bytes.
`0b0101`	32 bytes.
`0b0110`	64 bytes.
`0b0111`	128 bytes.
`0b1000`	256 bytes.
`0b1001`	512 bytes.
`0b1010`	1KB.

Value	Meaning
`0b1011`	2KB.

All other values are reserved.

For more information, see 'Restrictions on the current write pointer'.

If this field is non-zero, then every record is a multiple of this size.

Access to this field is **RO**.

Accessing the PMBIDR_EL1

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, PMBIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b111

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMBIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return PMBIDR_EL1;
elsif PSTATE.EL == EL2 then
    return PMBIDR_EL1;
elsif PSTATE.EL == EL3 then
    return PMBIDR_EL1;
```

A2.1.17 PMBSR_EL1, Profiling Buffer Status/syndrome Register

The PMBSR_EL1 characteristics are:

Purpose

Provides syndrome information to software when the buffer is disabled because the management interrupt has been raised.

Attributes

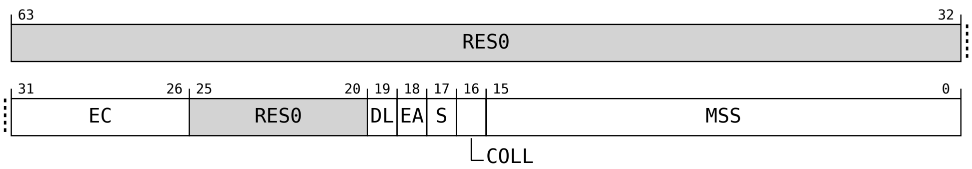
PMBSR_EL1 is a 64-bit register.

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMBSR_EL1 are UNDEFINED.

Field descriptions

The PMBSR_EL1 bit assignments are:



Bits [63:32]

Reserved, RES0.

EC, bits [31:26]

Exception class

Top-level description of the cause of the buffer management event

Value	Meaning	Link	Applies
0b000000	Other buffer management event. All buffer management events other than those described by other defined Exception class codes.	MSS - other buffer management events	
0b011110	Granule Protection Check fault, other than GPF, on write to Profiling Buffer.	MSS - Granule Protection Check fault	When FEAT_RME is implemented
0b011111	Buffer management event for an IMPLEMENTATION DEFINED reason.	MSS - a buffer management event for an IMPLEMENTATION DEFINED reason	
0b100100	Stage 1 Data Abort on write to Profiling Buffer.	MSS - stage 1 or stage 2 Data Aborts on write to buffer	
0b100101	Stage 2 Data Abort on write to Profiling Buffer.	MSS - stage 1 or stage 2 Data Aborts on write to buffer	

All other values are reserved. Reserved values might be defined in a future version of the architecture.

Writing a reserved value to this field will make the value of this field UNKNOWN. Values that are not supported act as reserved values when writing to this register.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [25:20]

Reserved, RES0.

DL, bit [19]

Partial record lost.

Following a buffer management event other than an asynchronous External abort, indicates whether the last record written to the Profiling Buffer is complete.

Value	Meaning
`0b0`	PMBPTR_EL1 points to the first byte after the last complete record written to the Profiling Buffer.
`0b1`	Part of a record was lost because of a buffer management event or synchronous External abort. PMBPTR_EL1 might not point to the first byte after the last complete record written to the buffer, and so restarting collection might result in a data record stream that software cannot parse. All records prior to the last record have been written to the buffer.

When the buffer management event was because of an asynchronous External abort, this bit is set to 1 and software must not assume that any valid data has been written to the Profiling Buffer.

This bit is RES0 if the PE never sets this bit as a result of a buffer management event caused by an asynchronous External abort.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [18]

External abort.

Value	Meaning
`0b0`	An External abort has not been asserted.
`0b1`	An External abort has been asserted and detected by the Statistical Profiling Extension.

This bit is RES0 if the PE never sets this bit as the result of an External abort.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S, bit [17]

Service

Value	Meaning
0b0	PMBIRQ is not asserted.
0b1	PMBIRQ is asserted. All profiling data has either been written to the buffer or discarded.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COLL, bit [16]

Collision detected.

Value	Meaning
0b0	No collision events detected.
0b1	At least one collision event was recorded.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

MSS, bits [15:0]

Management Event Specific Syndrome.

Contains syndrome specific to the management event.

stage 1 or stage 2 Data Aborts on write to buffer

15 – 6	5 – 0
RES0	FSC

Bits [15:6]

Reserved, RES0.

FSC, bits [5:0]

Fault status code

Value	Meaning	Applies
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	

Value	Meaning	Applies
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Asynchronous External abort.	
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b100001	Alignment fault.	
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented

Value	Meaning	Applies
`0b100110`	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
`0b100111`	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
`0b101000`	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
`0b101001`	Address size fault, level -1.	When FEAT_LPA2 is implemented
`0b101011`	Translation fault, level -1.	When FEAT_LPA2 is implemented
`0b110000`	TLB conflict abort.	
`0b110001`	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

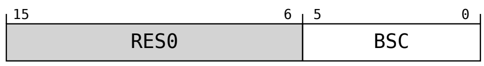
All other values are reserved.

It is IMPLEMENTATION DEFINED whether each of the Access Flag fault, asynchronous External abort and synchronous External abort, Alignment fault, and TLB Conflict abort values can be generated by the PE. For more information see 'Faults and Watchpoints'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

other buffer management events



Bits [15:6]

Reserved, RES0.

BSC, bits [5:0]

Buffer status code

Value	Meaning
`0b000000`	Buffer not filled
`0b000001`	Buffer filled

All other values are reserved. Reserved values might be defined in a future version of the architecture.

Writing a reserved value to this field will make the value of this field UNKNOWN. Values that are not supported act as reserved values when writing to this register.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Granule Protection Check fault

15 … 0
RES0

Bits [15:0]

Reserved, RES0.

a buffer management event for an IMPLEMENTATION DEFINED reason

15 … 0
IMPLEMENTATION DEFINED

IMPLEMENTATION DEFINED, bits [15:0]

IMPLEMENTATION DEFINED

The syndrome contents for each management event are described in the following sections.

Accessing the PMBSR_EL1

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, PMBSR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b011

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS || (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMBSR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2PB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS || (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        return NVMem[0x820];
    else
        return PMBSR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS || (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS || (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMBSR_EL1;
elsif PSTATE.EL == EL3 then
    return PMBSR_EL1;
```

MSR PMBSR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b011

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
     ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMBSR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2PB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
     ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        NVMem[0x820] = X[t];
    else
        PMBSR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
     ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
     ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMBSR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    PMBSR_EL1 = X[t];
```

A2.1.18 PMCCFILTR_EL0, Performance Monitors Cycle Count Filter Register

The PMCCFILTR_EL0 characteristics are:

Purpose

Determines the modes in which the Cycle Counter, PMCCNTR_EL0, increments.

Attributes

PMCCFILTR_EL0 is a 64-bit register.

Configuration

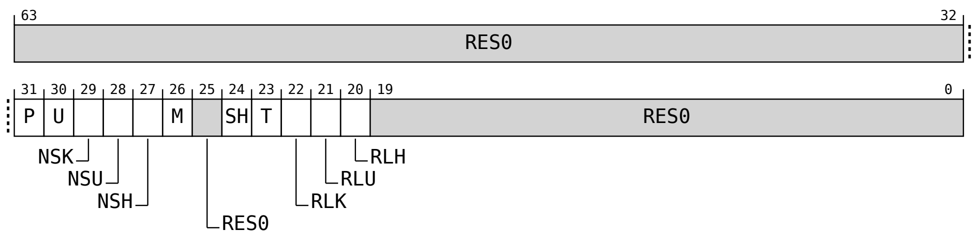
AArch64 system register PMCCFILTR_EL0 bits [31:0] are architecturally mapped to AArch32 system register PMCCFILTR[31:0].

AArch64 system register PMCCFILTR_EL0 bits [31:0] are architecturally mapped to External register PMCCFILTR_EL0[31:0].

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCCFILTR_EL0 are UNDEFINED.

Field descriptions

The PMCCFILTR_EL0 bit assignments are:



Bits [63:32]

Reserved, RES0.

P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMCCFILTR_EL0.NSK bit.

If FEAT_RME is implemented, then counting in Realm EL1 is further controlled by the PMCCFILTR_EL0.RLK bit.

Value	Meaning
`0b0`	Count cycles in EL1.
`0b1`	Do not count cycles in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMCCFILTR_EL0.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMCCFILTR_EL0.RLU bit.

Value	Meaning
`0b0`	Count cycles in EL0.
`0b1`	Do not count cycles in EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSK, bit [29]

When EL3 is implemented:

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Non-secure EL1 are counted.

Otherwise, cycles in Non-secure EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NSU, bit [28]

When EL3 is implemented:

Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Non-secure EL0 are counted.

Otherwise, cycles in Non-secure EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NSH, bit [27]

When EL2 is implemented:

EL2 (Hypervisor) filtering bit. Controls counting in EL2.

If Secure EL2 is implemented, and EL3 is implemented, counting in Secure EL2 is further controlled by the PMCCFILTR_EL0.SH bit.

If FEAT_RME is implemented, then counting in Realm EL2 is further controlled by the PMCCFILTR_EL0.RLH bit.

Value	Meaning
0b0	Do not count cycles in EL2.
0b1	Count cycles in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

M, bit [26]

When EL3 is implemented:

Secure EL3 filtering bit.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Secure EL3 are counted.

Otherwise, cycles in Secure EL3 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bit [25]

Reserved, RES0.

SH, bit [24]

When FEAT_SEL2 is implemented and EL3 is implemented:

Secure EL2 filtering.

If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Secure EL2 are counted.

Otherwise, cycles in Secure EL2 are not counted.

If Secure EL2 is disabled, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

T, bit [23]

When FEAT_TME is implemented:

Non-transactional state filtering bit.

Value	Meaning
0b0	This bit has no effect on filtering of cycles.
0b1	Do not count cycles in Non-transactional state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

RLK, bit [22]

When FEAT_RME is implemented:

Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Realm EL1 are counted.

Otherwise, cycles in Realm EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

RLU, bit [21]

When FEAT_RME is implemented:

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Realm EL0 are counted.

Otherwise, cycles in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

RLH, bit [20]

When FEAT_RME is implemented:

Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Realm EL2 are counted.

Otherwise, cycles in Realm EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [19:0]

Reserved, RES0.

Accessing the PMCCFILTR_EL0

PMCCFILTR_EL0 can also be accessed by using PMXEVTYPER_EL0 with PMSELR_EL0.SEL set to 0b11111.

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, PMCCFILTR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1111	0b111

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMCCFILTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCCFILTR_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMCCFILTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCCFILTR_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCCFILTR_EL0;
elsif PSTATE.EL == EL3 then
    return PMCCFILTR_EL0;
```

MSR PMCCFILTR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1111	0b111

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
      ↪when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
      ↪HDFGWTR_EL2.PMCCFILTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCCFILTR_EL0 = X[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
      ↪when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCCFILTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCCFILTR_EL0 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
      ↪when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCCFILTR_EL0 = X[t];
elsif PSTATE.EL == EL3 then
    PMCCFILTR_EL0 = X[t];
```

A2.1.19 PMEVTYPER<n>_EL0, Performance Monitors Event Type Registers, n = 0 - 30

The PMEVTYPER<n>_EL0 characteristics are:

Purpose

Configures event counter n, where n is 0 to 30.

Attributes

PMEVTYPER<n>_EL0 is a 64-bit register.

Configuration

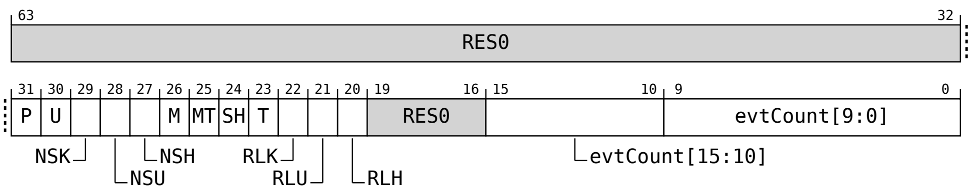
AArch64 system register PMEVTYPER<n>_EL0 bits [31:0] are architecturally mapped to AArch32 system register PMEVTYPER<n>[31:0].

AArch64 system register PMEVTYPER<n>_EL0 bits [31:0] are architecturally mapped to External register PMEVTYPER<n>_EL0[31:0].

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMEVTYPER<n>_EL0 are UNDEFINED.

Field descriptions

The PMEVTYPER<n>_EL0 bit assignments are:



Bits [63:32]

Reserved, RES0.

P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit.

If FEAT_RME is implemented, then counting in Realm EL1 is further controlled by the PMEVTYPER<n>_EL0.RLK bit.

Value	Meaning
`0b0`	Count events in EL1.
`0b1`	Do not count events in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMEVTYPER<n>_EL0.RLU bit.

Value	Meaning
`0b0`	Count events in EL0.
`0b1`	Do not count events in EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSK, bit [29]

When EL3 is implemented:

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted.

Otherwise, events in Non-secure EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NSU, bit [28]

When EL3 is implemented:

Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted.

Otherwise, events in Non-secure EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NSH, bit [27]

When EL2 is implemented:

EL2 (Hypervisor) filtering bit. Controls counting in EL2.

If Secure EL2 is implemented, and EL3 is implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit.

If FEAT_RME is implemented, then counting in Realm EL2 is further controlled by the PMEVTYPER<n>_EL0.RLH bit.

Value	Meaning
`0b0`	Do not count events in EL2.
`0b1`	Count events in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

M, bit [26]

When EL3 is implemented:

EL3 filtering bit.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted.

Otherwise, events in EL3 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

MT, bit [25]

When FEAT_MTPMU is implemented or an IMPLEMENTATION DEFINED multi-threaded PMU extension is implemented:

Multithreading.

Value	Meaning
`0b0`	Count events only on controlling PE.
`0b1`	Count events from any PE with the same affinity at level 1 and above as this PE.

From Armv8.6, the IMPLEMENTATION DEFINED multi-threaded PMU extension is not permitted, meaning if FEAT_MTPMU is not implemented, this field is RES0. See ID_AA64DFR0_EL1.MTPMU.

This field is ignored by the PE and treated as zero when FEAT_MTPMU is implemented and Disabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SH, bit [24]

When FEAT_SEL2 is implemented and EL3 is implemented:

Secure EL2 filtering.

If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted.

Otherwise, events in Secure EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

T, bit [23]

When FEAT_TME is implemented:

Transactional state filtering bit. Controls counting in Transactional state.

Value	Meaning
0b0	This bit has no effect on the filtering of events.
0b1	Do not count events in Transactional state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

RLK, bit [22]

When FEAT_RME is implemented:

Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted.

Otherwise, events in Realm EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

RLU, bit [21]

When FEAT_RME is implemented:

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted.

Otherwise, events in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

RLH, bit [20]

When FEAT_RME is implemented:

Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted.

Otherwise, events in Realm EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [19:16]

Reserved, RES0.

evtCount[15:10], bits [15:10]

When FEAT_PMUv3p1 is implemented:

Extension to evtCount[9:0]. For more information, see evtCount[9:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

evtCount[9:0], bits [9:0]

Event to count.

The event number of the event that is counted by event counter PMEVCNTR<n>_EL0.

The ranges of event numbers allocated to each type of event are shown in 'Allocation of the PMU event number space'.

If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written:

- For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.
- If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.
- For other values, it is UNPREDICTABLE what event, if any, is counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is UNKNOWN.

UNPREDICTABLE means the event must not expose privileged information.

Arm recommends that for all values that represent reserved or unsupported events, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing the PMEVTYPER<n>_EL0

PMEVTYPER<n>_EL0 can also be accessed by using PMXEVTYPER_EL0 with PMSELR_EL0.SEL set to n.

If FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <n> is greater than or equal to the number of accessible event counters, then reads and writes of PMEVTYPER<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

In EL0, an access is permitted if it is enabled by PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see MDCR_EL2.HPMN.

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, PMEVTYPER<n>_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b11:n[4:3]	n[2:0]

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    ↪when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
    ↪HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMEVTYPER_EL0[UInt(CRm<1:0>:op2<2:0>)];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    ↪when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMEVTYPER_EL0[UInt(CRm<1:0>:op2<2:0>)];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    ↪when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMEVTYPER_EL0[UInt(CRm<1:0>:op2<2:0>)];
elsif PSTATE.EL == EL3 then
    return PMEVTYPER_EL0[UInt(CRm<1:0>:op2<2:0>)];
```

MSR PMEVTYPER<n>_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b11:n[4:3]	n[2:0]

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    ↪when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
```

```
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
     ↪HDFGWTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[UInt(CRm<1:0>:op2<2:0>)] = X[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[UInt(CRm<1:0>:op2<2:0>)] = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[UInt(CRm<1:0>:op2<2:0>)] = X[t];
elsif PSTATE.EL == EL3 then
    PMEVTYPER_EL0[UInt(CRm<1:0>:op2<2:0>)] = X[t];
```

A2.1.20 SCR_EL3, Secure Configuration Register

The SCR_EL3 characteristics are:

Purpose

Defines the configuration of the current Security state. It specifies:

- The Security state of EL0, EL1, and EL2. The Security state is Secure, Non-secure, or Realm.
- The Execution state at lower Exception levels.
- Whether IRQ, FIQ, SError interrupts, and External abort exceptions are taken to EL3.
- Whether various operations are trapped to EL3.

Attributes

SCR_EL3 is a 64-bit register.

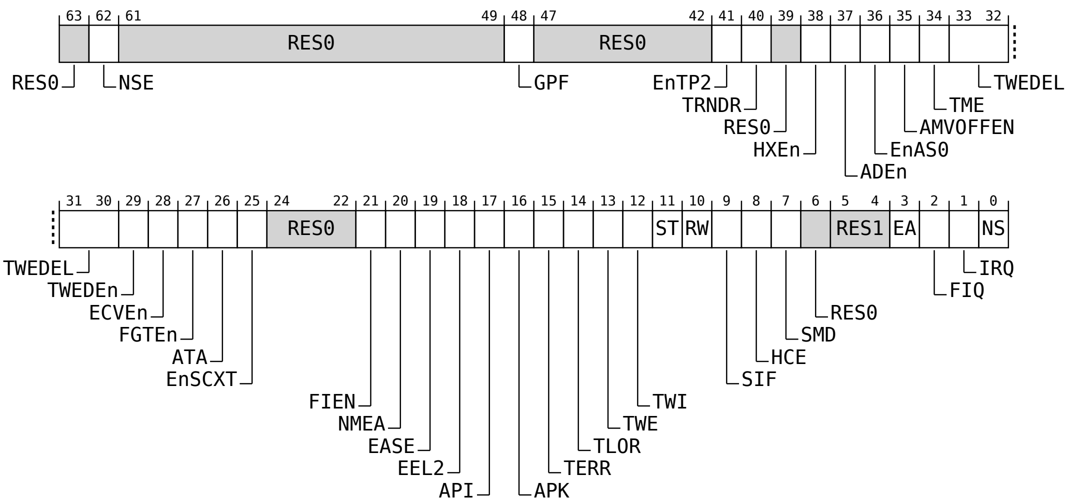
Configuration

AArch64 system register SCR_EL3 bits [31:0] can be mapped to AArch32 system register SCR[31:0], but this is not architecturally mandated.

This register is present only when EL3 is implemented. Otherwise, direct accesses to SCR_EL3 are UNDEFINED.

Field descriptions

The SCR_EL3 bit assignments are:



Bit [63]

Reserved, RES0.

NSE, bit [62]

When FEAT_RME is implemented:

This field, evaluated with SCR_EL3.NS, selects the Security state of EL2 and lower Exception levels.

For a description of the values derived by evaluating NS and NSE together, see SCR_EL3.NS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [61:49]

Reserved, RES0.

GPF, bit [48]

When FEAT_RME is implemented:

Controls the reporting of Granule protection faults at EL0, EL1 and EL2.

Value	Meaning
`0b0`	This control does not cause exceptions to be routed from EL0, EL1 or EL2 to EL3.
`0b1`	GPFs at EL0, EL1 and EL2 are routed to EL3 and reported as Granule Protection Check exceptions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [47:42]

Reserved, RES0.

EnTP2, bit [41]

When FEAT_SME is implemented:

Traps instructions executed at EL2, EL1, and EL0 that access TPIDR2_EL0 to EL3. The exception is reported using ESR_ELx.EC value 0x18.

Value	Meaning
`0b0`	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.
`0b1`	This control does not cause execution of any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TRNDR, bit [40]

When FEAT_RNG_TRAP is implemented:

Controls trapping of reads of RNDR and RNDRRS. The exception is reported using ESR_ELx.EC value 0x18.

Value	Meaning
`0b0`	This control does not cause RNDR and RDNRRS to be trapped. When FEAT_RNG is implemented: • ID_AA64ISAR0_EL1.RNDR returns the value 0b0001. When FEAT_RNG is not implemented: • ID_AA64ISAR0_EL1.RNDR returns the value 0b0000. • MRS reads of RNDR and RDNRRS are UNDEFINED.
`0b1`	ID_AA64ISAR0_EL1.RNDR returns the value 0b0001. Any attempt to read RNDR or RNDRRS is trapped to EL3.

When FEAT_RNG is not implemented, Arm recommends that SCR_EL3.TRNDR is initialized before entering Exception levels below EL3 and not subsequently changed.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

Bit [39]

Reserved, RES0.

HXEn, bit [38]

When FEAT_HCX is implemented:

Enables access to the HCRX_EL2 register at EL2 from EL3.

Value	Meaning
`0b0`	Accesses at EL2 to HCRX_EL2 are trapped to EL3. Indirect reads of HCRX_EL2 return 0.
`0b1`	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

ADEn, bit [37]

When FEAT_LS64 is implemented:

Enables access to the ACCDATA_EL1 register at EL1 and EL2.

Value	Meaning
`0b0`	Accesses to ACCDATA_EL1 at EL1 and EL2 are trapped to EL3, unless the accesses are trapped to EL2 by the EL2 fine-grained trap.
`0b1`	This control does not cause accesses to ACCDATA_EL1 to be trapped.

If the HFGWTR_EL2.nACCDATA_EL1 or HFGRTR_EL2.nACCDATA_EL1 traps are enabled, they take priority over this trap.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EnAS0, bit [36]

When FEAT_LS64 is implemented:

Traps execution of an ST64BV0 instruction at EL0, EL1, or EL2 to EL3.

Value	Meaning
`0b0`	EL0 execution of an ST64BV0 instruction is trapped to EL3, unless it is trapped to EL1 by SCTLR_EL1.EnAS0, or to EL2 by either HCRX_EL2.EnAS0 or SCTLR_EL2.EnAS0. EL1 execution of an ST64BV0 instruction is trapped to EL3, unless it is trapped to EL2 by HCRX_EL2.EnAS0. EL2 execution of an ST64BV0 instruction is trapped to EL3.
`0b1`	This control does not cause any instructions to be trapped.

A trap of an ST64BV0 instruction is reported using an ESR_ELx.EC value of 0x0A, with an ISS code of 0x0000001.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

AMVOFFEN, bit [35]

When FEAT_AMUv1p1 is implemented:

Activity Monitors Virtual Offsets Enable.

Value	Meaning
`0b0`	Accesses to AMEVCNTVOFF0<n>_EL2 and AMEVCNTVOFF1<n>_EL2 at EL2 are trapped to EL3. Indirect reads of the virtual offset registers are zero.
`0b1`	Accesses to AMEVCNTVOFF0<n>_EL2 and AMEVCNTVOFF1<n>_EL2 are not affected by this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TME, bit [34]

When FEAT_TME is implemented:

Enables access to the TSTART, TCOMMIT, TTEST and TCANCEL instructions at EL0, EL1 and EL2.

Value	Meaning
`0b0`	EL0, EL1 and EL2 accesses to TSTART, TCOMMIT, TTEST and TCANCEL instructions are UNDEFINED.
`0b1`	This control does not cause any instruction to be UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TWEDEL, bits [33:30]

When FEAT_TWED is implemented:

TWE Delay. A 4-bit unsigned number that, when SCR_EL3.TWEDEn is 1, encodes the minimum delay in taking a trap of WFE* caused by SCR_EL3.TWE as 2(TWEDEL + 8) cycles.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TWEDEn, bit [29]

When FEAT_TWED is implemented:

TWE Delay Enable. Enables a configurable delayed trap of the WFE* instruction caused by SCR_EL3.TWE.

Traps are reported using an ESR_ELx.EC value of 0x01.

Value	Meaning
`0b0`	The delay for taking the trap is IMPLEMENTATION DEFINED.
`0b1`	The delay for taking the trap is at least the number of cycles defined in SCR_EL3.TWEDEL.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

ECVEn, bit [28]

When FEAT_ECV is implemented:

ECV Enable. Enables access to the CNTPOFF_EL2 register.

Value	Meaning
`0b0`	EL2 accesses to CNTPOFF_EL2 are trapped to EL3, and the value of CNTPOFF_EL2 is treated as 0 for all purposes other than direct reads or writes to the register from EL3.
`0b1`	EL2 accesses to CNTPOFF_EL2 are not trapped to EL3 by this mechanism.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

FGTEn, bit [27]

When FEAT_FGT is implemented:

Fine-Grained Traps Enable. When EL2 is implemented, enables the traps to EL2 controlled by HAFGRTR_EL2, HDFGRTR_EL2, HDFGWTR_EL2, HFGRTR_EL2, HFGITR_EL2, and HFGWTR_EL2, and controls access to those registers.

If EL2 is not implemented but EL3 is implemented, FEAT_FGT implements the MDCR_EL3.TDCC traps.

Value	Meaning
`0b0`	EL2 accesses to HAFGRTR_EL2, HDFGRTR_EL2, HDFGWTR_EL2, HFGRTR_EL2, HFGITR_EL2 and HFGWTR_EL2 registers are trapped to EL3, and the traps to EL2 controlled by those registers are disabled.

Value	Meaning
0b1	EL2 accesses to HAFGRTR_EL2, HDFGRTR_EL2, HDFGWTR_EL2, HFGRTR_EL2, HFGITR_EL2 and HFGWTR_EL2 registers are not trapped to EL3 by this mechanism.

Traps caused by accesses to the fine-grained trap registers are reported using an ESR_ELx.EC value of 0x18 and its associated ISS.

Otherwise:

RES0

ATA, bit [26]

When FEAT_MTE2 is implemented:

Allocation Tag Access. Controls access at EL2, EL1 and EL0 to Allocation Tags.

Value	Meaning
0b0	Access to Allocation Tags is prevented. Accesses at EL1 and EL2 to GCR_EL1, RGSR_EL1, TFSR_EL1, TFSR_EL2 or TFSRE0_EL1 that are not UNDEFINED or trapped to a lower Exception level are trapped to EL3. Accesses at EL2 to rTFSR_EL12 that are not UNDEFINED are trapped to EL3.
0b1	This control does not prevent access to Allocation Tags.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

EnSCXT, bit [25]

When FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented:

Enable access to the SCXTNUM_EL2, SCXTNUM_EL1, and SCXTNUM_EL0 registers.

Value	Meaning
0b0	Accesses at EL0, EL1 and EL2 to SCXTNUM_EL0, SCXTNUM_EL1, or SCXTNUM_EL2 registers are trapped to EL3 if they are not trapped by a higher priority exception, and the values of these registers are treated as 0.
0b1	This control does not cause any accesses to be trapped, or register values to be treated as 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [24:22]

Reserved, RES0.

FIEN, bit [21]

When FEAT_RASv1p1 is implemented:

Fault Injection enable. Trap accesses to the registers ERXPFGCDN_EL1, ERXPFGCTL_EL1, and ERXPFGF_EL1 from EL1 and EL2 to EL3, reported using an ESR_ELx.EC value of 0x18.

Value	Meaning
`0b0`	Accesses to the specified registers from EL1 and EL2 generate a Trap exception to EL3.
`0b1`	This control does not cause any instructions to be trapped.

If EL3 is not implemented, the Effective value of SCR_EL3.FIEN is 0b1.

If ERRIDR_EL1.NUM is zero, meaning no error records are implemented, or no error record accessible using System registers is owned by a node that implements the RAS Common Fault Injection Model Extension, then this bit might be RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NMEA, bit [20]

When FEAT_DoubleFault is implemented:

Non-maskable External Aborts. When SCR_EL3.EA == 1, controls whether PSTATE.A masks SError interrupts at EL3.

Value	Meaning
`0b0`	If SCR_EL3.EA == 1, asserted SError interrupts are not taken at EL3 if PSTATE.A == 1.
`0b1`	If SCR_EL3.EA == 1, asserted SError interrupts are taken at EL3 regardless of the value of PSTATE.A.

When SCR_EL3.EA == 0:

- Asserted SError interrupts are not taken at EL3 regardless of the value of PSTATE.A and this field.
- This field is ignored and its Effective value is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

EASE, bit [19]

When FEAT_DoubleFault is implemented:

External aborts to SError interrupt vector.

Value	Meaning
`0b0`	Synchronous External abort exceptions taken to EL3 are taken to the appropriate synchronous exception vector offset from VBAR_EL3.
`0b1`	Synchronous External abort exceptions taken to EL3 are taken to the appropriate SError interrupt vector offset from VBAR_EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

EEL2, bit [18]

When FEAT_SEL2 is implemented:

Secure EL2 Enable.

Value	Meaning
`0b0`	All behaviors associated with Secure EL2 are disabled. All registers, including timer registers, defined by FEAT_SEL2 are UNDEFINED, and those timers are disabled.
`0b1`	All behaviors associated with Secure EL2 are enabled.

When the value of this bit is 1, then:

- When SCR_EL3.NS == 0, the SCR_EL3.RW bit is treated as 1 for all purposes other than reading or writing the register.
- If Secure EL1 is using AArch32, then any of the following operations, executed in Secure EL1, is trapped to Secure EL2, using the EC value of ESR_EL2.EC== 0x3 :
  - A read or write of the SCR.
  - A read or write of the NSACR.
  - A read or write of the MVBAR.
  - A read or write of the SDCR.
  - Execution of an ATS12NSO** instruction.
- If Secure EL1 is using AArch32, then any of the following operations, executed in Secure EL1, is trapped to

Secure EL2 using the EC value of ESR_EL2.EC== 0x0 :

- Execution of an SRS instruction that uses R13_mon.
- Execution of an MRS (Banked register) or MSR (Banked register) instruction that would access SPSR_mon, R13_mon, or R14_mon.

If the Effective value of SCR_EL3.EEL2 is 0, then these operations executed in Secure EL1 using AArch32 are trapped to EL3.

A Secure only implementation that does not implement EL3 but implements EL2, behaves as if SCR_EL3.EEL2 == 1.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bit [17]

When FEAT_SEL2 is implemented and FEAT_PAuth is implemented

API, bit [17]

Controls the use of the following instructions related to Pointer Authentication. Traps are reported using an ESR_ELx.EC value of 0x09:

- PACGA, which is always enabled.
- AUTDA, AUTDB, AUTDZA, AUTDZB, AUTIA, AUTIA1716, AUTIASP, AUTIAZ, AUTIB, AUTIB1716, AUTIBSP, AUTIBZ, AUTIZA, AUTIZB, PACDA, PACDB, PACDZA, PACDZB, PACIA, PACIA1716, PACIASP, PACIAZ, PACIB, PACIB1716, PACIBSP, PACIBZ, PACIZA, PACIZB, RETAA, RETAB, BRAA, BRAB, BLRAA, BLRAB, BRAAZ, BRABZ, BLRAAZ, BLRABZ, ERETAA, ERETAB, LDRAA and LDRAB when:
  - In EL0, when HCR_EL2.TGE == 0 or HCR_EL2.E2H == 0, and the associated SCTLR_EL1.En<N><M> == 1.
  - In EL0, when HCR_EL2.TGE == 1 and HCR_EL2.E2H == 1, and the associated SCTLR_EL2.En<N><M> == 1.
  - In EL1, when the associated SCTLR_EL1.En<N><M> == 1.
  - In EL2, when the associated SCTLR_EL2.En<N><M> == 1.

Value	Meaning
`0b0`	The use of any instruction related to pointer authentication in any Exception level except EL3 when the instructions are enabled are trapped to EL3 unless they are trapped to EL2 as a result of the HCR_EL2.API bit.
`0b1`	This control does not cause any instructions to be trapped.

An instruction is trapped only if Pointer Authentication is enabled for that instruction, for more information, see ‘System register control of pointer authentication’.

If FEAT_PAuth is implemented but EL3 is not implemented, the system behaves as if this bit is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_SEL2 is not implemented and FEAT_PAuth is implemented

API, bit [17]

Controls the use of instructions related to Pointer Authentication:

- PACGA.
- AUTDA, AUTDB, AUTDZA, AUTDZB, AUTIA, AUTIA1716, AUTIASP, AUTIAZ, AUTIB, AUTIB1716, AUTIBSP, AUTIBZ, AUTIZA, AUTIZB, PACDA, PACDB, PACDZA, PACDZB, PACIA, PACIA1716, PACIASP, PACIAZ, PACIB, PACIB1716, PACIBSP, PACIBZ, PACIZA, PACIZ, RETAA, RETAB, BRAA, BRAB, BLRAA, BLRAB, BRAAZ, BRABZ, BLRAAZ, BLRABZ, ERETAA, ERETAB, LDRAA and LDRAB when:
  - In Non-secure EL0, when HCR_EL2.TGE == 0 or HCR_EL2.E2H == 0, and the associated SCTLR_EL1.En<N><M>== 1.
  - In Non-secure EL0, when HCR_EL2.TGE == 1 and HCR_EL2.E2H == 1, and the associated SCTLR_EL2.En<N><M> == 1.
  - In Secure EL0, when the associated SCTLR_EL1.En<N><M> == 1.
  - In Secure or Non-secure EL1, when the associated SCTLR_EL1.En<N><M> == 1.
  - In EL2, when the associated SCTLR_EL2.En<N><M> == 1.

Value	Meaning
`0b0`	The use of any instruction related to pointer authentication in any Exception level except EL3 when the instructions are enabled are trapped to EL3 unless they are trapped to EL2 as a result of the HCR_EL2.API bit.
`0b1`	This control does not cause any instructions to be trapped.

If FEAT_PAuth is implemented but EL3 is not implemented, the system behaves as if this bit is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

APK, bit [16]

When FEAT_PAuth is implemented:

Trap registers holding "key" values for Pointer Authentication. Traps accesses to the following registers, using an ESR_ELx.EC value of 0x18, from EL1 or EL2 to EL3 unless they are trapped to EL2 as a result of the HCR_EL2.APK bit or other traps:

- APIAKeyLo_EL1, APIAKeyHi_EL1, APIBKeyLo_EL1, APIBKeyHi_EL1.
- APDAKeyLo_EL1, APDAKeyHi_EL1, APDBKeyLo_EL1, APDBKeyHi_EL1.
- APGAKeyLo_EL1, and APGAKeyHi_EL1.

Value	Meaning
`0b0`	Access to the registers holding "key" values for pointer authentication from EL1 or EL2 are trapped to EL3 unless they are trapped to EL2 as a result of the HCR_EL2.APK bit or other traps.

Value	Meaning
`0b1`	This control does not cause any instructions to be trapped.

For more information, see 'System register control of pointer authentication'.

If FEAT_PAuth is implemented but EL3 is not implemented, the system behaves as if this bit is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TERR, bit [15]

When FEAT_RAS is implemented:

Trap Error record accesses. Accesses to the RAS ERR* and RAS ERX* registers from EL1 and EL2 to EL3 are trapped as follows:

- Accesses from EL1 and EL2 using AArch64 to the following registers are trapped and reported using an ESR_ELx.EC value of 0x18:
  - ERRIDR_EL1, ERRSELR_EL1, ERXADDR_EL1, ERXCTLR_EL1, ERXFR_EL1, ERXMISC0_EL1, ERXMISC1_EL1, and ERXSTATUS_EL1.
- If FEAT_RASv1p1 is implemented, accesses from EL1 and EL2 using AArch64 to ERXMISC2_EL1, and ERXMISC3_EL1, are trapped and reported using an ESR_ELx.EC value of 0x18.
- Accesses from EL1 and EL2 using AArch32, to the following registers are trapped and reported using an ESR_ELx.EC value of 0x03:
  - ERRIDR, ERRSELR, ERXADDR, ERXADDR2, ERXCTLR, ERXCTLR2, ERXFR, ERXFR2, ERXMISC0, ERXMISC1, ERXMISC2, ERXMISC3, and ERXSTATUS.
- If FEAT_RASv1p1 is implemented, accesses from EL1 and EL2 using AArch32 to the following registers are trapped and reported using an ESR_ELx.EC value of 0x03:
  - ERXMISC4, ERXMISC5, ERXMISC6, and ERXMISC7.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	Accesses to the specified registers from EL1 and EL2 generate a Trap exception to EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TLOR, bit [14]

When FEAT_LOR is implemented:

Trap LOR registers. Traps accesses to the LORSA_EL1, LOREA_EL1, LORN_EL1, LORC_EL1, and LORID_EL1 registers from EL1 and EL2 to EL3, unless the access has been trapped to EL2.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	EL1 and EL2 accesses to the LOR registers that are not UNDEFINED are trapped to EL3, unless it is trapped HCR_EL2.TLOR.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

TWE, bit [13]

Traps EL2, EL1, and EL0 execution of WFE instructions to EL3, from any Security state and both Execution states, reported using an ESR_ELx.EC value of 0x01.

When FEAT_WFxT is implemented, this trap also applies to the WFET instruction.

Value	Meaning
`0b0`	This control does not cause any instructions to be trapped.
`0b1`	Any attempt to execute a WFE instruction at any Exception level lower than EL3 is trapped to EL3, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by SCTLR.nTWE, HCR.TWE, SCTLR_EL1.nTWE, SCTLR_EL2.nTWE, or HCR_EL2.TWE.

In AArch32 state, the attempted execution of a conditional WFE instruction is only trapped if the instruction passes its condition code check.

Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE of WFI are not guaranteed to be taken, even if the WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

For more information about when WFE instructions can cause the PE to enter a low-power state, see 'Wait for Event mechanism and Send event'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TWI, bit [12]

Traps EL2, EL1, and EL0 execution of WFI instructions to EL3, from any Security state and both Execution states, reported using an ESR_ELx.EC value of 0x01.

When FEAT_WFxT is implemented, this trap also applies to the WFIT instruction.

Value	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt to execute a WFI instruction at any Exception level lower than EL3 is trapped to EL3, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by SCTLR.nTWI, HCR.TWI, SCTLR_EL1.nTWI, SCTLR_EL2.nTWI, or HCR_EL2.TWI.

In AArch32 state, the attempted execution of a conditional WFI instruction is only trapped if the instruction passes its condition code check.

Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE of WFI are not guaranteed to be taken, even if the WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

For more information about when WFI instructions can cause the PE to enter a low-power state, see 'Wait for Interrupt'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ST, bit [11]

Traps Secure EL1 accesses to the Counter-timer Physical Secure timer registers to EL3, from AArch64 state only, reported using an ESR_ELx.EC value of 0x18.

Value	Meaning
0b0	Secure EL1 using AArch64 accesses to the CNTPS_TVAL_EL1, CNTPS_CTL_EL1, and CNTPS_CVAL_EL1 are trapped to EL3 when Secure EL2 is disabled. If Secure EL2 is enabled, the behavior is as if the value of this field was 0b1.
0b1	This control does not cause any instructions to be trapped.

Accesses to the Counter-timer Physical Secure timer registers are always enabled at EL3. These registers are not accessible at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

RW, bit [10]

When EL1 is capable of using AArch32 or EL2 is capable of using AArch32:

Execution state control for lower Exception levels.

Value	Meaning
0b0	Lower levels are all AArch32.

Value	Meaning
0b1	The next lower level is AArch64. If EL2 is present: • EL2 is AArch64. • EL2 controls EL1 and EL0 behaviors. If EL2 is not present: • EL1 is AArch64. • EL0 is determined by the Execution state described in the current process state when executing at EL0.

If AArch32 state is supported by the implementation at EL1, SCR_EL3.NS == 1 and AArch32 state is not supported by the implementation at EL2, the Effective value of this bit is 1.

If AArch32 state is supported by the implementation at EL1, FEAT_SEL2 is implemented and SCR_EL3.{EEL2, NS} == {1, 0}, the Effective value of this bit is 1.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RAO/WI

Bit [9]

When FEAT_SEL2 is implemented

SIF, bit [9]

Secure instruction fetch. When the PE is in Secure state, this bit disables instruction fetch from memory marked in the first stage of translation as being Non-secure.

Value	Meaning
0b0	Secure state instruction fetches from memory marked in the first stage of translation as being Non-secure are permitted.
0b1	Secure state instruction fetches from memory marked in the first stage of translation as being Non-secure are not permitted.

When FEAT_PAN3 is implemented, it is IMPLEMENTATION DEFINED whether SCR_EL3.SIF is also used to determine instruction access permission for the purpose of PAN.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

SIF, bit [9]

Secure instruction fetch. When the PE is in Secure state, this bit disables instruction fetch from Non-secure memory.

Value	Meaning
0b0	Secure state instruction fetches from Non-secure memory are permitted.
0b1	Secure state instruction fetches from Non-secure memory are not permitted.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

HCE, bit [8]

Hypervisor Call instruction enable. Enables HVC instructions at EL3 and, if EL2 is enabled in the current Security state, at EL2 and EL1, in both Execution states, reported using an ESR_ELx.EC value of 0x00.

Value	Meaning
0b0	HVC instructions are UNDEFINED.
0b1	HVC instructions are enabled at EL3, EL2, and EL1.

HVC instructions are always UNDEFINED at EL0 and, if Secure EL2 is disabled, at Secure EL1. Any resulting exception is taken from the current Exception level to the current Exception level.

If EL2 is not implemented, this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SMD, bit [7]

Secure Monitor Call disable. Disables SMC instructions at EL1 and above, from any Security state and both Execution states, reported using an ESR_ELx.EC value of 0x00.

Value	Meaning
0b0	SMC instructions are enabled at EL3, EL2 and EL1.
0b1	SMC instructions are UNDEFINED.

SMC instructions are always UNDEFINED at EL0. Any resulting exception is taken from the current Exception level to the current Exception level.

If HCR_EL2.TSC or HCR.TSC traps attempted EL1 execution of SMC instructions to EL2, that trap has priority over this disable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

Bits [5:4]

Reserved, RES1.

EA, bit [3]

External Abort and SError interrupt routing.

Value	Meaning
`0b0`	When executing at Exception levels below EL3, External aborts and SError interrupts are not taken to EL3. In addition, when executing at EL3: • SError interrupts are not taken. • External aborts are taken to EL3.
`0b1`	When executing at any Exception level, External aborts and SError interrupts are taken to EL3.

For more information, see 'Asynchronous exception routing'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FIQ, bit [2]

Physical FIQ Routing.

Value	Meaning
`0b0`	When executing at Exception levels below EL3, physical FIQ interrupts are not taken to EL3. When executing at EL3, physical FIQ interrupts are not taken.
`0b1`	When executing at any Exception level, physical FIQ interrupts are taken to EL3.

For more information, see 'Asynchronous exception routing'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IRQ, bit [1]

Physical IRQ Routing.

Value	Meaning
0b0	When executing at Exception levels below EL3, physical IRQ interrupts are not taken to EL3. When executing at EL3, physical IRQ interrupts are not taken.
0b1	When executing at any Exception level, physical IRQ interrupts are taken to EL3.

For more information, see 'Asynchronous exception routing'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [0]

When FEAT_RME is implemented

NS, bit [0]

Non-secure bit. This field is used in combination with SCR_EL3.NSE to select the Security state of EL2 and lower Exception levels.

NSE	NS	Meaning
0b0	0b0	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Reserved.
0b1	0b1	Realm.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise

NS, bit [0]

Non-secure bit.

Value	Meaning
0b0	Indicates that EL0 and EL1 are in Secure state.
0b1	Indicates that Exception levels lower than EL3 are in Non-secure state, so memory accesses from those Exception levels cannot access Secure memory.

When SCR_EL3.{EEL2, NS} == {1, 0}, then EL2 is using AArch64 and in Secure state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing the SCR_EL3

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, SCR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return SCR_EL3;
```

MSR SCR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SCR_EL3 = X[t];
```

A2.1.21 TRBIDR_EL1, Trace Buffer ID Register

The TRBIDR_EL1 characteristics are:

Purpose

Describes constraints on using the Trace Buffer Unit to software, including whether the Trace Buffer Unit can be programmed at the current Exception level.

Attributes

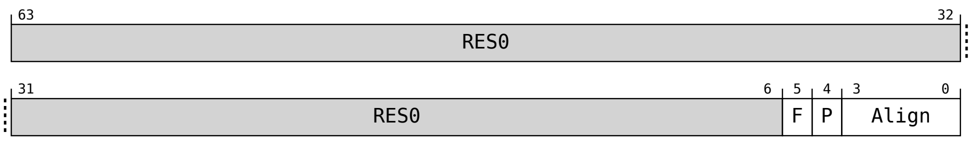
TRBIDR_EL1 is a 64-bit register.

Configuration

This register is present only when FEAT_TRBE is implemented. Otherwise, direct accesses to TRBIDR_EL1 are UNDEFINED.

Field descriptions

The TRBIDR_EL1 bit assignments are:



Bits [63:6]

Reserved, RES0.

F, bit [5]

Flag updates. Describes how address translations performed by the Trace Buffer Unit manage the Access flag and dirty state.

Value	Meaning
0b0	Hardware management of the Access flag and dirty state for accesses made by the Trace Buffer Unit is always disabled for all translation stages.
0b1	Hardware management of the Access flag and dirty state for accesses made by the Trace Buffer Unit is controlled in the same way as explicit memory accesses in the trace buffer owning translation regime.

If hardware management of the Access flag is disabled for a stage of translation, an access to a Page or Block with the Access flag bit not set in the descriptor will generate an Access Flag fault.

If hardware management of the dirty state is disabled for a stage of translation, an access to a Page or Block will ignore the Dirty Bit Modifier in the descriptor and might generate a Permission fault, depending on the values of the access permission bits in the descriptor.

Access to this field is **RO**.

P, bit [4]

Programming not allowed. When read at EL3, this field reads as zero. Otherwise, indicates that the trace buffer is owned by a higher Exception level or another Security state. Defined values are:

Value	Meaning
`0b0`	Programming is allowed.
`0b1`	Programming not allowed.

The value read from this field depends on the current Exception level and the Effective values of MDCR_EL3.NSTB, MDCR_EL3.NSTBE, and MDCR_EL2.E2TB:

- If EL3 is implemented, and the owning Security state is Secure state, this field reads as one from:
  - Non-secure EL1 and Non-secure EL2.
  - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
  - If Secure EL2 is implemented and enabled, and MDCR_EL2.E2TB is 0b00, Secure EL1.
- If EL3 is implemented, and the owning Security state is Non-secure state, this field reads as one from:
  - Secure EL1.
  - If Secure EL2 is implemented, Secure EL2.
  - If EL2 is implemented and MDCR_EL2.E2TB is 0b00, Non-secure EL1.
  - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
- If FEAT_RME is implemented, and the owning Security state is Realm state, this field reads as one from:
  - Non-secure EL1 and Non-secure EL2.
  - Secure EL1 and Secure EL2.
  - If MDCR_EL2.E2TB is 0b00, Realm EL1.
- If EL3 is not implemented, EL2 is implemented, and MDCR_EL2.E2TB is 0b00, this field reads as one from EL1.
- Otherwise, this field reads as zero.

Align, bits [3:0]

Defines the minimum alignment constraint for writes to TRBPTR_EL1 and TRBTRG_EL1. Defined values are:

Value	Meaning
`0b0000`	Byte.
`0b0001`	Halfword.
`0b0010`	Word.
`0b0011`	Doubleword.
`0b0100`	16 bytes.
`0b0101`	32 bytes.
`0b0110`	64 bytes.
`0b0111`	128 bytes.
`0b1000`	256 bytes.
`0b1001`	512 bytes.
`0b1010`	1KB.

Value	Meaning
`0b1011`	2KB.

All other values are reserved.

Access to this field is **RO**.

Accessing the TRBIDR_EL1

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, TRBIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b111

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.TRBIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return TRBIDR_EL1;
elsif PSTATE.EL == EL2 then
    return TRBIDR_EL1;
elsif PSTATE.EL == EL3 then
    return TRBIDR_EL1;
```

A2.1.22 TRBSR_EL1, Trace Buffer Status/syndrome Register

The TRBSR_EL1 characteristics are:

Purpose

Provides syndrome information to software for a trace buffer management event.

Attributes

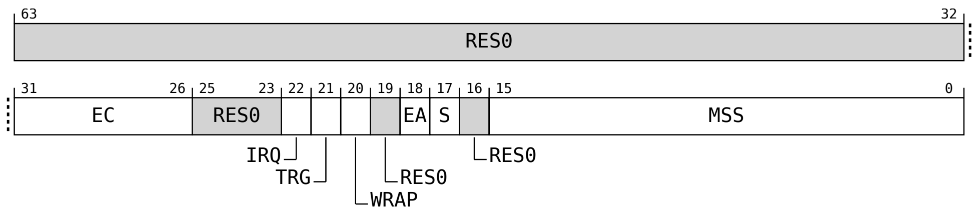
TRBSR_EL1 is a 64-bit register.

Configuration

This register is present only when FEAT_TRBE is implemented. Otherwise, direct accesses to TRBSR_EL1 are UNDEFINED.

Field descriptions

The TRBSR_EL1 bit assignments are:



Bits [63:32]

Reserved, RES0.

EC, bits [31:26]

Event class. Top-level description of the cause of the trace buffer management event.

Value	Meaning	Link	Applies
0b000000	Other trace buffer management event. All trace buffer management events other than those described by the other defined Event class codes.	MSS - other trace buffer management events	
0b011110	Granule Protection Check fault, other than GPF, on write to trace buffer.	MSS - Granule Protection Check fault	When FEAT_RME is implemented
0b011111	Buffer management event for IMPLEMENTATION DEFINED reason.	MSS - Buffer management event for IMPLEMENTATION DEFINED reason	
0b100100	Stage 1 Data Abort on write to trace buffer.	MSS - stage 1 or stage 2 Data Aborts on write to trace buffer	
0b100101	Stage 2 Data Abort on write to trace buffer.	MSS - stage 1 or stage 2 Data Aborts on write to trace buffer	

All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [25:23]

Reserved, RES0.

IRQ, bit [22]

Maintenance interrupt status.

Value	Meaning
`0b0`	Maintenance interrupt is not asserted.
`0b1`	Maintenance interrupt is asserted.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

TRG, bit [21]

Triggered.

Value	Meaning
`0b0`	No Detected Trigger has been observed since this field was last cleared to zero.
`0b1`	A Detected Trigger has been observed since this field was last cleared to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

WRAP, bit [20]

Wrapped.

Value	Meaning
`0b0`	The current write pointer has not wrapped since this field was last cleared to zero.
`0b1`	The current write pointer has wrapped since this field was last cleared to zero.

For each byte of trace the Trace Buffer Unit Accepts and writes to the trace buffer at the address in the current write pointer, if the current write pointer is equal to the Limit pointer minus one, the current write pointer is wrapped by setting it to the Base pointer, and this field is set to 1.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bit [19]

Reserved, RES0.

EA, bit [18]

External Abort.

Value	Meaning
`0b0`	An External Abort has not been asserted.
`0b1`	An External Abort has been asserted and detected by the Trace Buffer Unit.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When the PE never sets this field as the result of an External Abort, access is RES0
- Otherwise access is RW

S, bit [17]

Stopped.

Value	Meaning
`0b0`	Collection has not been stopped.
`0b1`	Collection is stopped.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

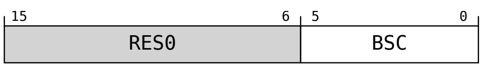
Bit [16]

Reserved, RES0.

MSS, bits [15:0]

Management Event Specific Syndrome. Contains syndrome specific to the management event.

other trace buffer management events



Bits [15:6]

Reserved, RES0.

BSC, bits [5:0]

Trace buffer status code.

Value	Meaning
0b000000	Collection not stopped.
0b000001	Trace buffer filled. Collection stopped because the current write pointer wrapped to the base pointer and the trace buffer mode is Fill mode.
0b000010	Trigger Event. Collection stopped because of a Trigger Event. See TRBTRG_EL1 for more information.

All other values are reserved.

Buffer management event for IMPLEMENTATION DEFINED reason

15 — 0
IMPLEMENTATION DEFINED

IMPLEMENTATION DEFINED, bits [15:0]

IMPLEMENTATION DEFINED

Granule Protection Check fault

15 — 0
RES0

Bits [15:0]

Reserved, RES0.

stage 1 or stage 2 Data Aborts on write to trace buffer

15 — 6	5 — 0
RES0	FSC

Bits [15:6]

Reserved, RES0.

FSC, bits [5:0]

Fault status code.

Value	Meaning	Applies
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	

Value	Meaning	Applies
`0b000110`	Translation fault, level 2.	
`0b000111`	Translation fault, level 3.	
`0b001001`	Access flag fault, level 1.	
`0b001010`	Access flag fault, level 2.	
`0b001011`	Access flag fault, level 3.	
`0b001000`	Access flag fault, level 0.	When FEAT_LPA2 is implemented
`0b001100`	Permission fault, level 0.	When FEAT_LPA2 is implemented
`0b001101`	Permission fault, level 1.	
`0b001110`	Permission fault, level 2.	
`0b001111`	Permission fault, level 3.	
`0b010000`	Synchronous External abort, not on translation table walk or hardware update of translation table.	
`0b010001`	Asynchronous External abort.	
`0b010011`	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
`0b010100`	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
`0b010101`	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
`0b010110`	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
`0b010111`	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
`0b011011`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
`0b100001`	Alignment fault.	
`0b100011`	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
`0b100100`	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
`0b100101`	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
`0b100110`	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
`0b100111`	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented

Value	Meaning	Applies
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

The syndrome contents for each management event are described in the following sections.

Accessing the TRBSR_EL1

The PE might ignore a direct write to TRBSR_EL1 if TRBLIMITR_EL1.E == 1.

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, TRBSR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b011

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
      ↪when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
      ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.TRBSR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
      ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRBSR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
      ↪when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
      ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
      ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRBSR_EL1;
elsif PSTATE.EL == EL3 then
    return TRBSR_EL1;
```

MSR TRBSR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b011

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
     ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.TRBSR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
     ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRBSR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
     ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
     ↪(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRBSR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    TRBSR_EL1 = X[t];
```

A2.1.23 TRCAUTHSTATUS, Authentication Status Register

The TRCAUTHSTATUS characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

For additional information, see the CoreSight Architecture Specification.

Attributes

TRCAUTHSTATUS is a 64-bit register.

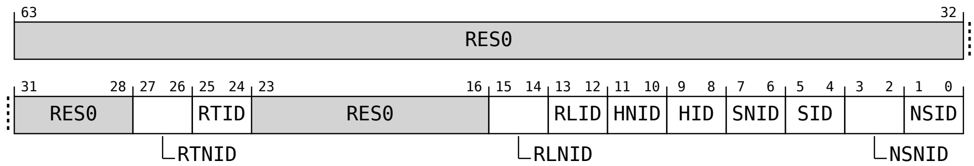
Configuration

AArch64 system register TRCAUTHSTATUS bits [31:0] are architecturally mapped to External register TRCAUTHSTATUS[31:0].

This register is present only when FEAT_ETE is implemented. Otherwise, direct accesses to TRCAUTHSTATUS are UNDEFINED.

Field descriptions

The TRCAUTHSTATUS bit assignments are:



Bits [63:28]

Reserved, RES0.

RTNID, bits [27:26]

Root non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RTNID.

RTID, bits [25:24]

Root invasive debug.

Value	Meaning
`0b00`	Not implemented.

Bits [23:16]

Reserved, RES0.

RLNID, bits [15:14]

Realm non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RLNID.

RLID, bits [13:12]

Realm invasive debug.

Value	Meaning
`0b00`	Not implemented.

HNID, bits [11:10]

Hyp Non-invasive Debug. Indicates whether a separate enable control for EL2 non-invasive debug features is implemented and enabled.

Value	Meaning
`0b00`	Separate Hyp non-invasive debug enable not implemented, or EL2 non-invasive debug features not implemented.
`0b10`	Implemented and disabled.
`0b11`	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

HID, bits [9:8]

Hyp Invasive Debug. Indicates whether a separate enable control for EL2 invasive debug features is implemented and enabled.

Value	Meaning
`0b00`	Separate Hyp invasive debug enable not implemented, or EL2 invasive debug features not implemented.
`0b10`	Implemented and disabled.
`0b11`	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

SNID, bits [7:6]

Secure Non-invasive Debug. Indicates whether Secure non-invasive debug features are implemented and enabled.

Value	Meaning
`0b00`	Secure non-invasive debug features not implemented.
`0b10`	Implemented and disabled.

Value	Meaning
`0b11`	Implemented and enabled.

All other values are reserved.

When EL3 is implemented, this field takes the value 0b10 or 0b11 depending whether Secure non-invasive debug is enabled.

When EL3 is not implemented and the PE is Non-secure, this field reads as 0b00.

When EL3 is not implemented and the PE is Secure, this field takes the value 0b10 or 0b11 depending whether Secure non-invasive debug is enabled.

SID, bits [5:4]

Secure Invasive Debug. Indicates whether Secure invasive debug features are implemented and enabled.

Value	Meaning
`0b00`	Secure invasive debug features not implemented.
`0b10`	Implemented and disabled.
`0b11`	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

NSNID, bits [3:2]

Non-secure Non-invasive Debug. Indicates whether Non-secure non-invasive debug features are implemented and enabled.

Value	Meaning
`0b00`	Non-secure non-invasive debug features not implemented.
`0b10`	Implemented and disabled.
`0b11`	Implemented and enabled.

All other values are reserved.

When EL3 is implemented, this field reads as 0b11.

When EL3 is not implemented and the PE is Non-secure, this field reads as 0b11.

When EL3 is not implemented and the PE is Secure, this field reads as 0b00.

NSID, bits [1:0]

Non-secure Invasive Debug. Indicates whether Non-secure invasive debug features are implemented and enabled.

Value	Meaning
0b00	Non-secure invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

Accessing the TRCAUTHSTATUS

For implementations that support multiple access mechanisms, different access mechanisms can return different values for reads of TRCAUTHSTATUS if the authentication signals have changed and that change has not yet been synchronized by a Context synchronization event. This scenario can happen if, for example, the external debugger view is implemented separately from the system instruction view to allow for separate power domains, and so observes changes on the signals differently.

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, TRCAUTHSTATUS

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1110	0b110

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
        ↪when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.TRCAUTHSTATUS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCAUTHSTATUS;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
        ↪when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCAUTHSTATUS;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCAUTHSTATUS;
```

A2.2 GIC registers

A2.2.1 ICC_CTLR_EL3, Interrupt Controller Control Register (EL3)

The ICC_CTLR_EL3 characteristics are:

Purpose

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Attributes

ICC_CTLR_EL3 is a 64-bit register.

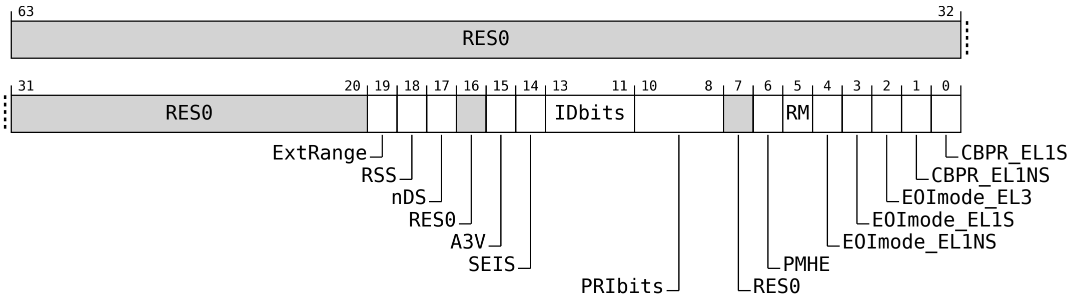
Configuration

AArch64 system register ICC_CTLR_EL3 bits [31:0] can be mapped to AArch32 system register ICC_MCTLR[31:0], but this is not architecturally mandated.

This register is present only when FEAT_GICv3 is implemented and EL3 is implemented. Otherwise, direct accesses to ICC_CTLR_EL3 are UNDEFINED.

Field descriptions

The ICC_CTLR_EL3 bit assignments are:



Bits [63:20]

Reserved, RES0.

ExtRange, bit [19]

Extended INTID range (read-only).

Value	Meaning
`0b0`	CPU interface does not support INTIDs in the range 1024..8191. • Behaviour is UNPREDICTABLE if the IRI delivers an interrupt in the range 1024 to 8191 to the CPU interface. Arm strongly recommends that the IRI is not configured to deliver interrupts in this range to a PE that does not support them.
`0b1`	CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.

RSS, bit [18]

Range Selector Support.

Value	Meaning
`0b0`	Targeted SGIs with affinity level 0 values of 0-15 are supported.
`0b1`	Targeted SGIs with affinity level 0 values of 0-255 are supported.

This bit is read-only.

nDS, bit [17]

Disable Security not supported. Read-only and writes are ignored.

Value	Meaning
`0b0`	The CPU interface logic supports disabling of security.
`0b1`	The CPU interface logic does not support disabling of security, and requires that security is not disabled.

When a PE implements the Realm Management Extension, this field is RAO/WI.

Bit [16]

Reserved, RES0.

A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored.

Value	Meaning
`0b0`	The CPU interface logic does not support non-zero values of the Aff3 field in SGI generation System registers.
`0b1`	The CPU interface logic supports non-zero values of the Aff3 field in SGI generation System registers.

If EL3 is present, ICC_CTLR_EL1.A3V is an alias of ICC_CTLR_EL3.A3V

SEIS, bit [14]

SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports generation of SEIs:

Value	Meaning
`0b0`	The CPU interface logic does not support generation of SEIs.

Value	Meaning
`0b1`	The CPU interface logic supports generation of SEIs.

If EL3 is present, ICC_CTLR_EL1.SEIS is an alias of ICC_CTLR_EL3.SEIS

IDbits, bits [13:11]

Identifier bits. Read-only and writes are ignored. Indicates the number of physical interrupt identifier bits supported.

Value	Meaning
`0b000`	16 bits.
`0b001`	24 bits.

All other values are reserved.

If EL3 is present, ICC_CTLR_EL1.IDbits is an alias of ICC_CTLR_EL3.IDbits

PRIbits, bits [10:8]

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).

An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).

This field always returns the number of priority bits implemented, regardless of the value of SCR_EL3.NS or the value of GICD_CTLR.DS.

The division between group priority and subpriority is defined in the binary point registers ICC_BPR0_EL1 and ICC_BPR1_EL1.

This field determines the minimum value of ICC_BPR0_EL1.

Bit [7]

Reserved, RES0.

PMHE, bit [6]

Priority Mask Hint Enable.

Value	Meaning
`0b0`	Disables use of the priority mask register as a hint for interrupt distribution.
`0b1`	Enables use of the priority mask register as a hint for interrupt distribution.

Software must write ICC_PMR_EL1 to 0xFF before clearing this field to 0.

- An implementation might choose to make this field RAO/WI if priority-based routing is always used
- An implementation might choose to make this field RAZ/WI if priority-based routing is never used

If EL3 is present, ICC_CTLR_EL1.PMHE is an alias of ICC_CTLR_EL3.PMHE.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

RM, bit [5]

Routing Modifier. This bit controls whether EL3 can acknowledge, or observe as the Highest Priority Pending Interrupt, Secure Group 0 and Non-secure Group 1 interrupts.

Value	Meaning
`0b0`	Secure Group 0 and Non-secure Group 1 interrupts can be acknowledged and observed as the highest priority interrupt at EL3.
`0b1`	Secure Group 0 and Non-secure Group 1 interrupts cannot be acknowledged and observed as the highest priority interrupt at EL3. Secure Group 0 interrupts return a special INTID value of 1020. This affects accesses to ICC_IAR0_EL1 and ICC_HPPIR0_EL1. Non-secure Group 1 interrupts return a special INTID value of 1021. This affects accesses to ICC_IAR1_EL1 and ICC_HPPIR1_EL1.

The Routing Modifier bit is supported in AArch64 only. In systems without EL3 the behavior is as if the value is 0. Software must ensure this bit is 0 when the Secure copy of ICC_SRE_EL1.SRE is 1, otherwise system behavior is UNPREDICTABLE. In systems without EL3 or where the Secure copy of ICC_SRE_EL1.SRE is RAO/WI, this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EOImode_EL1NS, bit [4]

EOI mode for interrupts handled at Non-secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt.

Value	Meaning
`0b0`	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are UNPREDICTABLE.
`0b1`	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality.

If EL3 is present, ICC_CTLR_EL1(NS).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1NS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EOImode_EL1S, bit [3]

EOI mode for interrupts handled at Secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt.

Value	Meaning
`0b0`	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are UNPREDICTABLE.
`0b1`	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality.

If EL3 is present, ICC_CTLR_EL1(S).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1S.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EOImode_EL3, bit [2]

EOI mode for interrupts handled at EL3. Controls whether a write to an End of Interrupt register also deactivates the interrupt.

Value	Meaning
`0b0`	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are UNPREDICTABLE.
`0b1`	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CBPR_EL1NS, bit [1]

Common Binary Point Register, EL1 Non-secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Non-secure interrupts at EL1 and EL2.

Value	Meaning
`0b0`	ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. ICC_BPR1_EL1 determines the preemption group for Non-secure Group 1 interrupts.

Value	Meaning
0b1	ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Non-secure Group 1 interrupts. Non-secure accesses to GICC_BPR and ICC_BPR1_EL1 access the state of ICC_BPR0_EL1.

If EL3 is present, ICC_CTLR_EL1(NS).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1NS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CBPR_EL1S, bit [0]

Common Binary Point Register, EL1 Secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Secure interrupts at EL1 and EL2.

Value	Meaning
0b0	ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. ICC_BPR1_EL1 determines the preemption group for Secure Group 1 interrupts.
0b1	ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Secure Group 1 interrupts. Secure EL1 accesses to ICC_BPR1_EL1 access the state of ICC_BPR0_EL1.

If EL3 is present, ICC_CTLR_EL1(S).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1S.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing the ICC_CTLR_EL3

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, ICC_CTLR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b1100	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_CTLR_EL3;
```

MSR ICC_CTLR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b1100	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_CTLR_EL3 = X[t];
```

A2.2.2 ICC_SRE_EL1, Interrupt Controller System Register Enable register (EL1)

The ICC_SRE_EL1 characteristics are:

Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL1.

Attributes

ICC_SRE_EL1 is a 64-bit register.

Configuration

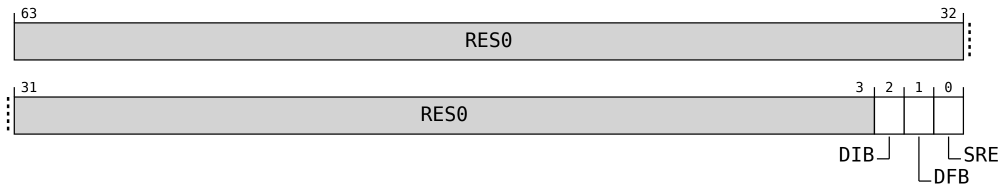
AArch64 system register ICC_SRE_EL1 bits 31:0 are architecturally mapped to AArch32 system register ICC_SRE31:0.

AArch64 system register ICC_SRE_EL1 bits 31:0 are architecturally mapped to AArch32 system register ICC_SRE31:0.

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_SRE_EL1 are UNDEFINED.

Field descriptions

The ICC_SRE_EL1 bit assignments are:



Bits [63:3]

Reserved, RES0.

DIB, bit [2]

Disable IRQ bypass.

Value	Meaning
`0b0`	IRQ bypass enabled.
`0b1`	IRQ bypass disabled.

If EL3 is implemented and GICD_CTLR.DS == 0, this field is a read-only alias of ICC_SRE_EL3.DIB.

If EL3 is implemented and GICD_CTLR.DS == 1, and EL2 is not implemented, this field is a read-write alias of ICC_SRE_EL3.DIB.

If EL3 is not implemented and EL2 is implemented, this field is a read-only alias of ICC_SRE_EL2.DIB.

If GICD_CTLR.DS == 1 and EL2 is implemented, this field is a read-only alias of ICC_SRE_EL2.DIB.

In systems that do not support IRQ bypass, this field is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

DFB, bit [1]

Disable FIQ bypass.

Value	Meaning
`0b0`	FIQ bypass enabled.
`0b1`	FIQ bypass disabled.

If EL3 is implemented and GICD_CTLR.DS == 0, this field is a read-only alias of ICC_SRE_EL3.DFB.

If EL3 is implemented and GICD_CTLR.DS == 1, and EL2 is not implemented, this field is a read-write alias of ICC_SRE_EL3.DFB.

If EL3 is not implemented and EL2 is implemented, this field is a read-only alias of ICC_SRE_EL2.DFB.

If GICD_CTLR.DS == 1 and EL2 is implemented, this field is a read-only alias of ICC_SRE_EL2.DFB.

In systems that do not support FIQ bypass, this field is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

SRE, bit [0]

System Register Enable.

Value	Meaning
`0b0`	The memory-mapped interface must be used. Access at EL1 to any ICC_* System register other than ICC_SRE_EL1 is trapped to EL1.
`0b1`	The System register interface for the current Security state is enabled.

If software changes this bit from 1 to 0 in the Secure instance of this register, the results are UNPREDICTABLE.

If an implementation supports only a System register interface to the GIC CPU interface, this bit is RAO/WI.

If EL3 is implemented and ICC_SRE_EL3.SRE==0 the Secure copy of this bit is RAZ/WI. If ICC_SRE_EL3.SRE is changed from zero to one, the Secure copy of this bit becomes UNKNOWN.

If EL2 is implemented and ICC_SRE_EL2.SRE==0 the Non-secure copy of this bit is RAZ/WI. If ICC_SRE_EL2.SRE is changed from zero to one, the Non-secure copy of this bit becomes UNKNOWN.

If EL3 is implemented and ICC_SRE_EL3.SRE==0 the Non-secure copy of this bit is RAZ/WI. If ICC_SRE_EL3.SRE is changed from zero to one, the Non-secure copy of this bit becomes UNKNOWN.

If Realm Management Extension is implemented, this field is RAO/WI.

GICv3 implementations that do not require GICv2 compatibility might choose to make this bit RAO/WI. The following options are supported:

- The Non-secure copy of ICC_SRE_EL1.SRE can be RAO/WI if ICC_SRE_EL2.SRE is also RAO/WI. This means all Non-secure software, including VMs using only virtual interrupts, must access the GIC using

System registers.

- The Secure copy of ICC_SRE_EL1.SRE can be RAO/WI if ICC_SRE_EL3.SRE and ICC_SRE_EL2.SRE are also RAO/WI. This means that all Secure software must access the GIC using System registers and all Non-secure accesses to registers for physical interrupts must use System registers.

A VM using only virtual interrupts might still use memory-mapped access if the Non-secure copy of ICC_SRE_EL1.SRE is not RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Accessing the ICC_SRE_EL1

Execution with ICC_SRE_EL1.SRE set to 0 might make some System registers UNKNOWN.

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, ICC_SRE_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ICC_SRE_EL2.Enable == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            return ICC_SRE_EL1_S;
        else
            return ICC_SRE_EL1_NS;
    else
        return ICC_SRE_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            return ICC_SRE_EL1_S;
        else
            return ICC_SRE_EL1_NS;
    else
        return ICC_SRE_EL1;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        return ICC_SRE_EL1_S;
    else
        return ICC_SRE_EL1_NS;
```

MSR ICC_SRE_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    ↪when SDD == '1'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ICC_SRE_EL2.Enable == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_SRE_EL1_S = X[t];
        else
            ICC_SRE_EL1_NS = X[t];
    else
        ICC_SRE_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    ↪when SDD == '1'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_SRE_EL1_S = X[t];
        else
            ICC_SRE_EL1_NS = X[t];
    else
        ICC_SRE_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        ICC_SRE_EL1_S = X[t];
    else
        ICC_SRE_EL1_NS = X[t];
```

A2.2.3 ICC_SRE_EL2, Interrupt Controller System Register Enable register (EL2)

The ICC_SRE_EL2 characteristics are:

Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL2.

Attributes

ICC_SRE_EL2 is a 64-bit register.

Configuration

If EL2 is not implemented, this register is RES0 from EL3.

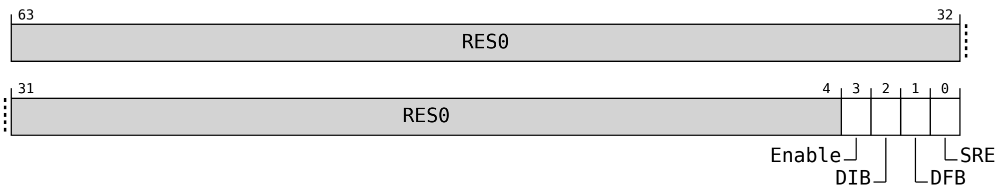
This register has no effect if EL2 is not enabled in the current Security state.

AArch64 system register ICC_SRE_EL2 bits [31:0] are architecturally mapped to AArch32 system register ICC_HSRE[31:0].

This register is present only when FEAT_GICv3 is implemented and (EL2 is implemented or EL3 is implemented). Otherwise, direct accesses to ICC_SRE_EL2 are UNDEFINED.

Field descriptions

The ICC_SRE_EL2 bit assignments are:



Bits [63:4]

Reserved, RES0.

Enable, bit [3]

Enable. Enables lower Exception level access to ICC_SRE_EL1.

Value	Meaning
`0b0`	When EL2 is implemented and enabled in the current Security state, EL1 accesses to ICC_SRE_EL1 trap to EL2.
`0b1`	EL1 accesses to ICC_SRE_EL1 do not trap to EL2.

If ICC_SRE_EL2.SRE is RAO/WI, an implementation is permitted to make the Enable bit RAO/WI.

If ICC_SRE_EL2.SRE is 0, the Enable bit behaves as 1 for all purposes other than reading the value of the bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DIB, bit [2]

Disable IRQ bypass.

Value	Meaning
0b0	IRQ bypass enabled.
0b1	IRQ bypass disabled.

If EL3 is implemented and GICD_CTLR.DS is 0, this field is a read-only alias of ICC_SRE_EL3.DIB.

If EL3 is implemented and GICD_CTLR.DS is 1, this field is a read-write alias of ICC_SRE_EL3.DIB.

In systems that do not support IRQ bypass, this bit is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0b0.

DFB, bit [1]

Disable FIQ bypass.

Value	Meaning
0b0	FIQ bypass enabled.
0b1	FIQ bypass disabled.

If EL3 is implemented and GICD_CTLR.DS is 0, this field is a read-only alias of ICC_SRE_EL3.DFB.

If EL3 is implemented and GICD_CTLR.DS is 1, this field is a read-write alias of ICC_SRE_EL3.DFB.

In systems that do not support FIQ bypass, this bit is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0b0.

SRE, bit [0]

System Register Enable.

Value	Meaning
0b0	The memory-mapped interface must be used. Access at EL2 to any ICH_* or ICC_* register other than ICC_SRE_EL1 or ICC_SRE_EL2, is trapped to EL2.
0b1	The System register interface to the ICH_* registers and the EL1 and EL2 ICC_* registers is enabled for EL2.

If software changes this bit from 1 to 0, the results are UNPREDICTABLE.

If an implementation supports only a System register interface to the GIC CPU interface, this bit is RAO/WI.

If EL3 is implemented and ICC_SRE_EL3.SRE==0 this bit is RAZ/WI. If ICC_SRE_EL3.SRE is changed from zero to one, this bit becomes UNKNOWN.

If Realm Management Extension is implemented, this field is RAO/WI.

FEAT_GICv3 implementations that do not require GICv2 compatibility might choose to make this bit RAO/WI, but this is only allowed if ICC_SRE_EL3.SRE is also RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Accessing the ICC_SRE_EL2

Execution with ICC_SRE_EL2.SRE set to 0 might make some System registers UNKNOWN.

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, ICC_SRE_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_SRE_EL2;
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    else
        return ICC_SRE_EL2;
```

MSR ICC_SRE_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
```

```
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
        ↪when SDD == '1'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_SRE_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    else
        ICC_SRE_EL2 = X[t];
```

A2.2.4 ICC_SRE_EL3, Interrupt Controller System Register Enable register (EL3)

The ICC_SRE_EL3 characteristics are:

Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL3.

Attributes

ICC_SRE_EL3 is a 64-bit register.

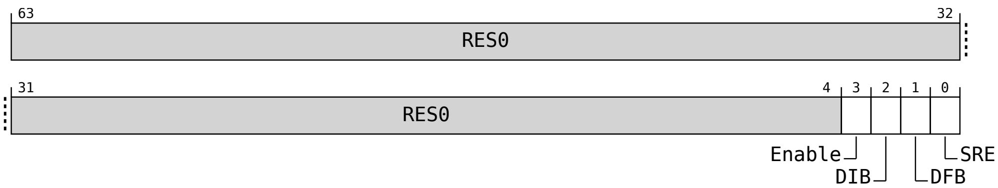
Configuration

AArch64 system register ICC_SRE_EL3 bits [31:0] can be mapped to AArch32 system register ICC_MSRE[31:0], but this is not architecturally mandated.

This register is present only when FEAT_GICv3 is implemented and EL3 is implemented. Otherwise, direct accesses to ICC_SRE_EL3 are UNDEFINED.

Field descriptions

The ICC_SRE_EL3 bit assignments are:



Bits [63:4]

Reserved, RES0.

Enable, bit [3]

Enable. Enables lower Exception level access to ICC_SRE_EL1 and ICC_SRE_EL2.

Value	Meaning
`0b0`	EL1 accesses to ICC_SRE_EL1 trap to EL3, unless these accesses are trapped to EL2 as a result of ICC_SRE_EL2.Enable == 0. EL2 accesses to ICC_SRE_EL1 and ICC_SRE_EL2 trap to EL3.
`0b1`	EL1 accesses to ICC_SRE_EL1 do not trap to EL3. EL2 accesses to ICC_SRE_EL1 and ICC_SRE_EL2 do not trap to EL3.

If ICC_SRE_EL3.SRE is RAO/WI, an implementation is permitted to make the Enable bit RAO/WI.

If ICC_SRE_EL3.SRE is 0, the Enable bit behaves as 1 for all purposes other than reading the value of the bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DIB, bit [2]

Disable IRQ bypass.

Value	Meaning
0b0	IRQ bypass enabled.
0b1	IRQ bypass disabled.

In systems that do not support IRQ bypass, this bit is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0b0.

DFB, bit [1]

Disable FIQ bypass.

Value	Meaning
0b0	FIQ bypass enabled.
0b1	FIQ bypass disabled.

In systems that do not support FIQ bypass, this bit is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0b0.

SRE, bit [0]

System Register Enable.

Value	Meaning
0b0	The memory-mapped interface must be used. Access at EL3 to any ICH_* or ICC_* register other than ICC_SRE_EL1, ICC_SRE_EL2, or ICC_SRE_EL3 is trapped to EL3
0b1	The System register interface to the ICH_* registers and the EL1, EL2, and EL3 ICC_* registers is enabled for EL3.

If software changes this bit from 1 to 0, the results are UNPREDICTABLE.

If Realm Management Extension is implemented, this field is RAO/WI.

FEAT_GICv3 implementations that do not require GICv2 compatibility might choose to make this bit RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0b0.

Accessing the ICC_SRE_EL3

This register is always System register accessible.

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, ICC_SRE_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b1100	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return ICC_SRE_EL3;
```

MSR ICC_SRE_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b1100	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    ICC_SRE_EL3 = X[t];
```

A2.2.5 ICH_VTR_EL2, Interrupt Controller VGIC Type Register

The ICH_VTR_EL2 characteristics are:

Purpose

Reports supported GIC virtualization features.

Attributes

ICH_VTR_EL2 is a 64-bit register.

Configuration

If EL2 is not implemented, all bits in this register are RES0 from EL3, except for nV4, which is RES1 from EL3.

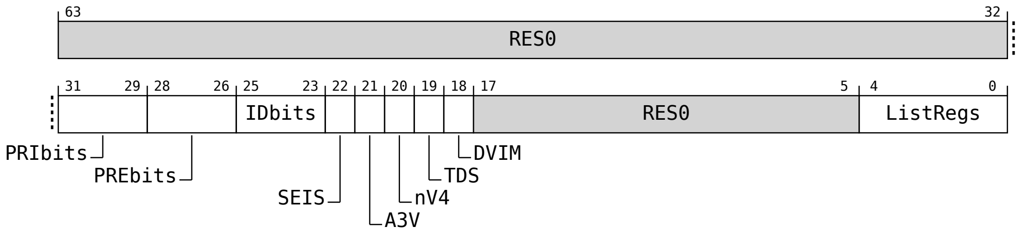
This register has no effect if EL2 is not enabled in the current Security state.

AArch64 system register ICH_VTR_EL2 bits [31:0] are architecturally mapped to AArch32 system register ICH_VTR[31:0].

This register is present only when FEAT_GICv3 is implemented and (EL2 is implemented or EL3 is implemented). Otherwise, direct accesses to ICH_VTR_EL2 are UNDEFINED.

Field descriptions

The ICH_VTR_EL2 bit assignments are:



Bits [63:32]

Reserved, RES0.

PRIbits, bits [31:29]

Priority bits. The number of virtual priority bits implemented, minus one.

An implementation must implement at least 32 levels of virtual priority (5 priority bits).

This field is an alias of ICV_CTLR_EL1.PRIbits.

PREbits, bits [28:26]

The number of virtual preemption bits implemented, minus one.

An implementation must implement at least 32 levels of virtual preemption priority (5 preemption bits).

The value of this field must be less than or equal to the value of ICH_VTR_EL2.PRIbits.

The maximum value of this field is 6, indicating 7 bits of preemption.

This field determines the minimum value of ICH_VMCR_EL2.VBPR0.

IDbits, bits [25:23]

The number of virtual interrupt identifier bits supported:

Value	Meaning
`0b000`	16 bits.
`0b001`	24 bits.

All other values are reserved.

This field is an alias of ICV_CTLR_EL1.IDbits.

SEIS, bit [22]

SEI Support. Indicates whether the virtual CPU interface supports generation of SEIs:

Value	Meaning
`0b0`	The virtual CPU interface logic does not support generation of SEIs.
`0b1`	The virtual CPU interface logic supports generation of SEIs.

This bit is an alias of ICV_CTLR_EL1.SEIS.

A3V, bit [21]

Affinity 3 Valid. Possible values are:

Value	Meaning
`0b0`	The virtual CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
`0b1`	The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

This bit is an alias of ICV_CTLR_EL1.A3V.

nV4, bit [20]

Direct injection of virtual interrupts not supported. Possible values are:

Value	Meaning
`0b0`	The CPU interface logic supports direct injection of virtual interrupts.
`0b1`	The CPU interface logic does not support direct injection of virtual interrupts.

In GICv3, the only permitted value is 0b1.

TDS, bit [19]

Separate trapping of EL1 writes to ICV_DIR_EL1 supported.

Value	Meaning
`0b0`	Implementation does not support ICH_HCR_EL2.TDIR.
`0b1`	Implementation supports ICH_HCR_EL2.TDIR.

FEAT_GICv3_TDIR implements the functionality added by the value 0b1.

DVIM, bit [18]

Masking of directly-injected virtual interrupts.

Value	Meaning
`0b0`	Masking of Directly-injected Virtual Interrupts not supported.
`0b1`	Masking of Directly-injected Virtual Interrupts is supported.

When a PE implements the Realm Management Extension, this field is RAO/WI.

Bits [17:5]

Reserved, RES0.

ListRegs, bits [4:0]

The number of implemented List registers, minus one. For example, a value of 0b01111 indicates that the maximum of 16 List registers are implemented.

Accessing the ICH_VTR_EL2

Accesses to this register use the following encodings in the instruction encoding space:

MRS <Xt>, ICH_VTR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1011	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ICH_VTR_EL2;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICH_VTR_EL2;
```

A2.3 AArch32 registers

A2.3.1 PMCCFILTR, Performance Monitors Cycle Count Filter Register

The PMCCFILTR characteristics are:

Purpose

Determines the modes in which the Cycle Counter, PMCCNTR, increments.

Attributes

PMCCFILTR is a 32-bit register.

Configuration

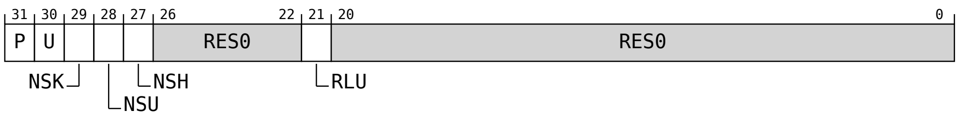
AArch32 system register PMCCFILTR bits [31:0] are architecturally mapped to AArch64 system register PMCCFILTR_EL0[31:0].

AArch32 system register PMCCFILTR bits [31:0] are architecturally mapped to External register PMCCFILTR_EL0[31:0].

This register is present only when AArch32 is supported and FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCCFILTR are UNDEFINED.

Field descriptions

The PMCCFILTR bit assignments are:



P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMCCFILTR.NSK bit.

Value	Meaning
`0b0`	Count cycles in EL1.
`0b1`	Do not count cycles in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMCCFILTR.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMCCFILTR.RLU bit.

Value	Meaning
`0b0`	Count cycles in EL0.

Value	Meaning
`0b1`	Do not count cycles in EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

NSK, bit [29]

When EL3 is implemented:

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of PMCCFILTR.P, cycles in Non-secure EL1 are counted.

Otherwise, cycles in Non-secure EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

NSU, bit [28]

When EL3 is implemented:

Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of PMCCFILTR.U, cycles in Non-secure EL0 are counted.

Otherwise, cycles in Non-secure EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

NSH, bit [27]

When EL2 is implemented:

EL2 (Hyp mode) filtering bit. Controls counting in EL2.

Value	Meaning
`0b0`	Do not count cycles in EL2.
`0b1`	Count cycles in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Otherwise:

RES0

Bits [26:22]

Reserved, RES0.

RLU, bit [21]

When FEAT_RME is implemented:

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMCCFILTR.U bit, cycles in Realm EL0 are counted.

Otherwise, cycles in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [20:0]

Reserved, RES0.

Accessing the PMCCFILTR

PMCCFILTR can also be accessed by using PMXEVTYPER with PMSELR.SEL set to 0b11111.

Accesses to this register use the following encodings in the instruction encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b1111	0b111

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMCCFILTR_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
```

```
    else
        return PMCCFILTR;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        return PMCCFILTR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        return PMCCFILTR;
elsif PSTATE.EL == EL3 then
    return PMCCFILTR;
```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b1111	0b111

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) ||
     ↪SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCCFILTR_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCCFILTR = R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
     ↪when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
```

```
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCCFILTR = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    ↪when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCCFILTR = R[t];
elsif PSTATE.EL == EL3 then
    PMCCFILTR = R[t];
```

A2.3.2 PMEVTYPER<n>, Performance Monitors Event Type Registers, n = 0 - 30

The PMEVTYPER<n> characteristics are:

Purpose

Configures event counter n, where n is 0 to 30.

Attributes

PMEVTYPER<n> is a 32-bit register.

Configuration

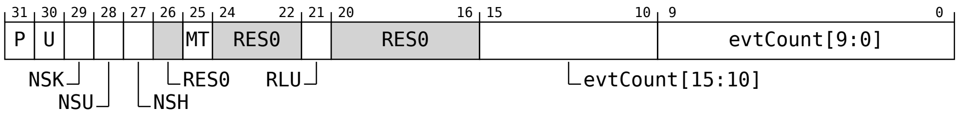
AArch32 system register PMEVTYPER<n> bits [31:0] are architecturally mapped to AArch64 system register PMEVTYPER<n>_EL0[31:0].

AArch32 system register PMEVTYPER<n> bits [31:0] are architecturally mapped to External register PMEVTYPER<n>_EL0[31:0].

This register is present only when AArch32 is supported and FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMEVTYPER<n> are UNDEFINED.

Field descriptions

The PMEVTYPER<n> bit assignments are:



P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>.NSK bit.

Value	Meaning
0b0	Count events in EL1.
0b1	Do not count events in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMEVTYPER<n>.RLU bit.

Value	Meaning
0b0	Count events in EL0.

Value	Meaning
`0b1`	Do not count events in EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSK, bit [29]

When EL3 is implemented:

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of PMEVTYPER<n>.P, events in Non-secure EL1 are counted.

Otherwise, events in Non-secure EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NSU, bit [28]

When EL3 is implemented:

Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of PMEVTYPER<n>.U, events in Non-secure EL0 are counted.

Otherwise, events in Non-secure EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NSH, bit [27]

When EL2 is implemented:

EL2 (Hyp mode) filtering bit. Controls counting in EL2.

Value	Meaning
`0b0`	Do not count events in EL2.
`0b1`	Count events in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bit [26]

Reserved, RES0.

MT, bit [25]

When FEAT_MTPMU is implemented or an IMPLEMENTATION DEFINED multi-threaded PMU extension is implemented:

Multithreading.

Value	Meaning
`0b0`	Count events only on controlling PE.
`0b1`	Count events from any PE with the same affinity at level 1 and above as this PE.

From Armv8.6, the IMPLEMENTATION DEFINED multi-threaded PMU extension is not permitted, meaning if FEAT_MTPMU is not implemented, this bit is RES0. See ID_DFR1.MTPMU.

This bit is ignored by the PE and treated as zero when FEAT_MTPMU is implemented and Disabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [24:22]

Reserved, RES0.

RLU, bit [21]

When FEAT_RME is implemented:

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>.U bit, events in Realm EL0 are counted.

Otherwise, events in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [20:16]

Reserved, RES0.

evtCount[15:10], bits [15:10]

When FEAT_PMUv3p1 is implemented:

Extension to evtCount[9:0]. For more information, see evtCount[9:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

evtCount[9:0], bits [9:0]

Event to count.

The event number of the event that is counted by event counter PMEVCNTR<n>.

The ranges of event numbers allocated to each type of event are shown in 'Allocation of the PMU event number space'.

If PMEVTYPER<n>.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written:

- For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>.evtCount field is the value written to the field.
- If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>.evtCount field is the value written to the field.
- For other values, it is UNPREDICTABLE what event, if any, is counted and the value returned by a direct or external read of the PMEVTYPER<n>.evtCount field is UNKNOWN.

UNPREDICTABLE means the event must not expose privileged information.

Arm recommends that for all values that represent reserved or unsupported events, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>.evtCount field is the value written to the field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing the PMEVTYPER<n>

PMEVTYPER<n> can also be accessed by using PMXEVTYPER with PMSELR.SEL set to n.

If FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of PMEVTYPER<n> is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <n> is greater than or equal to the number of accessible event counters, then reads and writes of PMEVTYPER<n> are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

In EL0, an access is permitted if it is enabled by PMUSERENR.EN or PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, at EL0 and EL1:

- If EL2 is using AArch32, HDCR.HPMN identifies the number of accessible event counters.
- If EL2 is using AArch64, MDCR_EL2.HPMN identifies the number of accessible event counters.

Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see HDCR.HPMN and MDCR_EL2.HPMN.

Accesses to this register use the following encodings in the instruction encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b11:n[4:3]	n[2:0]

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    ↪when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) ||
    ↪SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        return PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    ↪when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        return PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    ↪when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        return PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)];
elsif PSTATE.EL == EL3 then
    return PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)];
```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b11:n[4:3]	n[2:0]

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)] = R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)] = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)] = R[t];
elsif PSTATE.EL == EL3 then
    PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)] = R[t];
```

A2.4 External registers

A2.4.1 CTIAUTHSTATUS, CTI Authentication Status register

The CTIAUTHSTATUS characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for CTI.

Attributes

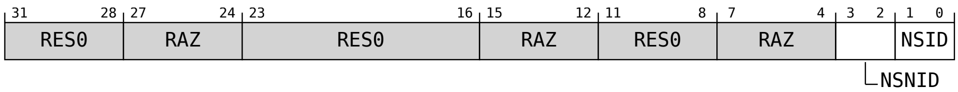
CTIAUTHSTATUS is a 32-bit register.

Configuration

This register is OPTIONAL, and is required for CoreSight compliance.

Field descriptions

The CTIAUTHSTATUS bit assignments are:



Bits [31:28]

Reserved, RES0.

Bits [27:24]

Reserved, RAZ.

Bits [23:16]

Reserved, RES0.

Bits [15:12]

Reserved, RAZ.

Bits [11:8]

Reserved, RES0.

Bits [7:4]

Reserved, RAZ.

NSNID, bits [3:2]

If EL3 is implemented, this field holds the same value as DBGAUTHSTATUS_EL1.NSNID.

If EL3 is not implemented and the implemented Security state is Secure state, this field holds the same value as DBGAUTHSTATUS_EL1.SNID.

NSID, bits [1:0]

If EL3 is implemented, this field holds the same value as DBGAUTHSTATUS_EL1.NSID.

If EL3 is not implemented and the implemented Security state is Secure state, this field holds the same value as DBGAUTHSTATUS_EL1.SID.

Accessing the CTIAUTHSTATUS

CTIAUTHSTATUS can be accessed through the external debug interface:

Component	Offset	Instance
CTI	0xFB8	CTIAUTHSTATUS

Access on this interface is **RO**.

A2.4.2 DBGAUTHSTATUS_EL1, Debug Authentication Status register

The DBGAUTHSTATUS_EL1 characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

Attributes

DBGAUTHSTATUS_EL1 is a 32-bit register.

Configuration

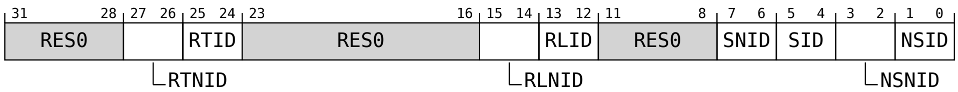
If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

External register DBGAUTHSTATUS_EL1 bits [31:0] are architecturally mapped to AArch64 system register DBGAUTHSTATUS_EL1[31:0].

External register DBGAUTHSTATUS_EL1 bits [31:0] are architecturally mapped to AArch32 system register DBGAUTHSTATUS[31:0].

Field descriptions

The DBGAUTHSTATUS_EL1 bit assignments are:



Bits [31:28]

Reserved, RES0.

RTNID, bits [27:26]

Root non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RTID.

RTID, bits [25:24]

Root invasive debug.

Value	Meaning
`0b00`	Not implemented.
`0b10`	Implemented and disabled. ExternalRootInvasiveDebugEnabled() == FALSE.
`0b11`	Implemented and enabled. ExternalRootInvasiveDebugEnabled() == TRUE.

All other values are reserved.

If FEAT_RME is not implemented, the only permitted value is 00.

Bits [23:16]

Reserved, RES0.

RLNID, bits [15:14]

Realm non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RLID.

RLID, bits [13:12]

Realm invasive debug.

Value	Meaning
`0b00`	Not implemented.
`0b10`	Implemented and disabled. ExternalRealmInvasiveDebugEnabled() == FALSE.
`0b11`	Implemented and enabled. ExternalRealmInvasiveDebugEnabled() == TRUE.

All other values are reserved.

If FEAT_RME is not implemented, the only permitted value is 00.

Bits [11:8]

Reserved, RES0.

Bits [7:6]

When FEAT_Debugv8p4 is implemented

SNID, bits [7:6]

Secure non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.SID.

Otherwise

SNID, bits [7:6]

Secure non-invasive debug.

Value	Meaning
`0b00`	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 1.
`0b10`	Implemented and disabled. ExternalSecureNoninvasiveDebugEnabled() == FALSE.
`0b11`	Implemented and enabled. ExternalSecureNoninvasiveDebugEnabled() == TRUE.

All other values are reserved.

SID, bits [5:4]

Secure invasive debug.

Value	Meaning
0b00	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 1.
0b10	Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.

All other values are reserved.

Bits [3:2]

When FEAT_Debugv8p4 is implemented

NSNID, bits [3:2]

Non-secure non-invasive debug.

Value	Meaning
0b00	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
0b11	Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE.

If the Effective value of SCR_EL3.NS is 1, or if EL3 is implemented and EL2 is not implemented, this field reads as 0b11.

All other values are reserved.

Otherwise

NSNID, bits [3:2]

Non-secure non-invasive debug.

Value	Meaning
0b00	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
0b10	Implemented and disabled. ExternalNoninvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE.

All other values are reserved.

NSID, bits [1:0]

Non-secure invasive debug.

Value	Meaning
`0b00`	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
`0b10`	Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE.
`0b11`	Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE.

All other values are reserved.

Accessing the DBGAUTHSTATUS_EL1

DBGAUTHSTATUS_EL1 can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0xFB8	DBGAUTHSTATUS_EL1

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered() access to this register is **RO**.
- Otherwise access to this register returns an ERROR.

A2.4.3 EDECCR, External Debug Exception Catch Control Register

The EDECCR characteristics are:

Purpose

Controls Exception Catch debug events. For more information, see ‘Summary of Exception Catch debug event control’.

Attributes

EDECCR is a 32-bit register.

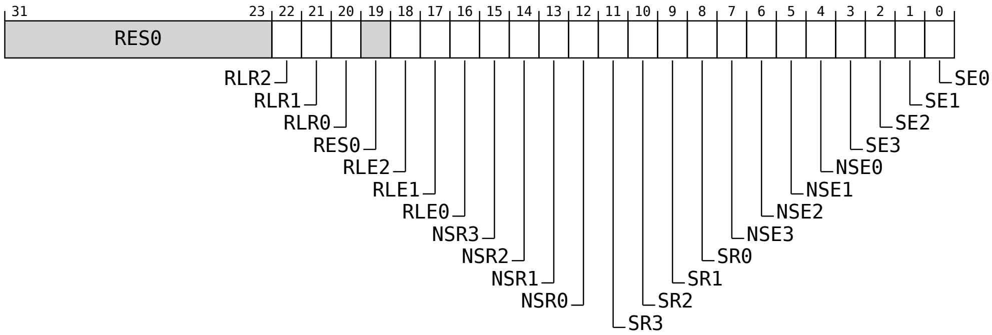
Configuration

External register EDECCR bits [31:0] are architecturally mapped to AArch64 system register OSECCR_EL1[31:0].

External register EDECCR bits [31:0] are architecturally mapped to AArch32 system register DBGOSECCR[31:0].

Field descriptions

The EDECCR bit assignments are:



Bits [31:23]

Reserved, RES0.

RLR2, bit [22]

When FEAT_RME is implemented:

Controls exception catch on exception return to Realm EL2 in conjunction with EDECCR.RLE2.

Value	Meaning
`0b0`	If EDECCR.RLE2 is 0, then Exception Catch debug events are disabled for Realm EL2. If EDECCR.RLE2 is 1, then Exception Catch debug events are enabled for exception entry and exception return to Realm EL2.
`0b1`	If EDECCR.RLE2 is 0, then Exception Catch debug events are enabled for exception returns to Realm EL2. If EDECCR.RLE2 is 1, then Exception Catch debug events are enabled for exception entry to Realm EL2.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

RLR1, bit [21]

When FEAT_RME is implemented:

Controls exception catch on exception return to Realm EL1 in conjunction with EDECCR.RLE1.

Value	Meaning
`0b0`	If EDECCR.RLE1 is 0, then Exception Catch debug events are disabled for Realm EL1. If EDECCR.RLE1 is 1, then Exception Catch debug events are enabled for exception entry and exception return to Realm EL1.
`0b1`	If EDECCR.RLE1 is 0, then Exception Catch debug events are enabled for exception returns to Realm EL1. If EDECCR.RLE1 is 1, then Exception Catch debug events are enabled for exception entry to Realm EL1.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

RLR0, bit [20]

When FEAT_RME is implemented:

Controls exception catch on exception return to Realm EL0.

Value	Meaning
`0b0`	Exception Catch debug events are disabled for Realm EL0.
`0b1`	Exception Catch debug events are enabled for exception returns to Realm EL0.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

Bit [19]

Reserved, RES0.

RLE2, bit [18]

When FEAT_RME is implemented:

Controls exception catch on exception entry to Realm EL2. Also controls exception catch on exception return to Realm EL2 in conjunction with EDECCR.RLR2.

Value	Meaning
`0b0`	If EDECCR.RLR2 is 0, then Exception Catch debug events are disabled for Realm EL2. If EDECCR.RLR2 is 1, then Exception Catch debug events are enabled for exception returns to Realm EL2.
`0b1`	If EDECCR.RLR2 is 0, then Exception Catch debug events are enabled for exception entry and exception return to Realm EL2. If EDECCR.RLR2 is 1, then Exception Catch debug events are enabled for exception entry to Realm EL2.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

RLE1, bit [17]

When FEAT_RME is implemented:

Controls exception catch on exception entry to Realm EL1. Also controls exception catch on exception return to Realm EL1 in conjunction with EDECCR.RLR1.

Value	Meaning
`0b0`	If EDECCR.RLR1 is 0, then Exception Catch debug events are disabled for Realm EL1. If EDECCR.RLR1 is 1, then Exception Catch debug events are enabled for exception returns to Realm EL1.
`0b1`	If EDECCR.RLR1 is 0, then Exception Catch debug events are enabled for exception entry and exception return to Realm EL1. If EDECCR.RLR1 is 1, then Exception Catch debug events are enabled for exception entry to Realm EL1.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

RLE0, bit [16]

Access to this field is **RES0**.

NSR3, bit [15]

Access to this field is **RES0**.

NSR2, bit [14]

When FEAT_Debugv8p2 is implemented and Non-secure EL2 is implemented:

Controls exception catch on exception return to Non-secure EL2 in conjunction with EDECCR.NSE2.

Value	Meaning
`0b0`	If EDECCR.NSE2 is 0, then Exception Catch debug events are disabled for Non-secure EL2. If EDECCR.NSE2 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Non-secure EL2.
`0b1`	If EDECCR.NSE2 is 0, then Exception Catch debug events are enabled for exception returns to Non-secure EL2. If EDECCR.NSE2 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Non-secure EL2.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

NSR1, bit [13]

When FEAT_Debugv8p2 is implemented and Non-secure EL1 is implemented:

Controls exception catch on exception return to Non-secure EL1 in conjunction with EDECCR.NSE1.

Value	Meaning
`0b0`	If EDECCR.NSE1 is 0, then Exception Catch debug events are disabled for Non-secure EL1. If EDECCR.NSE1 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Non-secure EL1.
`0b1`	If EDECCR.NSE1 is 0, then Exception Catch debug events are enabled for exception returns to Non-secure EL1. If EDECCR.NSE1 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Non-secure EL1.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

NSR0, bit [12]

When FEAT_Debugv8p2 is implemented and Non-secure EL0 is implemented:

Controls exception catch on exception return to Non-secure EL0.

Value	Meaning
`0b0`	Exception Catch debug events are disabled for Non-secure EL0.
`0b1`	Exception Catch debug events are enabled for exception returns to Non-secure EL0.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

SR3, bit [11]

When FEAT_Debugv8p2 is implemented and EL3 is implemented:

Controls exception catch on exception return to EL3 in conjunction with EDECCR.SE3.

Value	Meaning
`0b0`	If EDECCR.SE3 is 0, then Exception Catch debug events are disabled for EL3. If EDECCR.SE3 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to EL3.
`0b1`	If EDECCR.SE3 is 0, then Exception Catch debug events are enabled for exception returns to EL3. If EDECCR.SE3 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to EL3.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

SR2, bit [10]

When FEAT_Debugv8p2 is implemented and FEAT_SEL2 is implemented:

Controls exception catch on exception return to Secure EL2 in conjunction with EDECCR.SE2.

Value	Meaning
`0b0`	If EDECCR.SE2 is 0, then Exception Catch debug events are disabled for Secure EL2. If EDECCR.SE2 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL2.
`0b1`	If EDECCR.SE2 is 0, then Exception Catch debug events are enabled for exception returns to Secure EL2. If EDECCR.SE2 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL2.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

SR1, bit [9]

When FEAT_Debugv8p2 is implemented and Secure EL1 is implemented:

Controls exception catch on exception return to Secure EL1 in conjunction with EDECCR.SE1.

Value	Meaning
`0b0`	If EDECCR.SE1 is 0, then Exception Catch debug events are disabled for Secure EL1. If EDECCR.SE1 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL1.
`0b1`	If EDECCR.SE1 is 0, then Exception Catch debug events are enabled for exception returns to Secure EL1. If EDECCR.SE1 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL1.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

SR0, bit [8]

When FEAT_Debugv8p2 is implemented and Secure EL0 is implemented:

Controls exception catch on exception return to Secure EL0.

Value	Meaning
0b0	Exception Catch debug events are disabled for Secure EL0.
0b1	Exception Catch debug events are enabled for exception returns to Secure EL0.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0b0.

Otherwise:

RES0

NSE3, bit [7]

Access to this field is **RES0**.

Bit [6]

When FEAT_Debugv8p2 is implemented and Non-secure EL2 is implemented

NSE2, bit [6]

Controls exception catch on exception entry to Non-secure EL2. Also controls exception catch on exception return to Non-secure EL2 in conjunction with EDECCR.NSR2.

Value	Meaning
0b0	If EDECCR.NSR2 is 0, then Exception Catch debug events are disabled for Non-secure EL2. If EDECCR.NSR2 is 1, then Exception Catch debug events are enabled for exception returns to Non-secure EL2.
0b1	If EDECCR.NSR2 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Non-secure EL2. If EDECCR.NSR2 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Non-secure EL2.

It is IMPLEMENTATION DEFINED whether a reset entry to an Exception level will generate an Exception Catch debug event.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0b0.

When Non-secure EL2 is implemented

NSE2, bit [6]

Coarse-grained exception catch for Non-secure EL2. Controls Exception Catch debug events for Non-secure EL2.

Value	Meaning
0b0	Exception Catch debug events are disabled for Non-secure EL2.
0b1	Exception Catch debug events are enabled for Non-secure EL2.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0b0.

Otherwise:

RES0

Bit [5]

When FEAT_Debugv8p2 is implemented and Non-secure EL1 is implemented

NSE1, bit [5]

Controls exception catch on exception entry to Non-secure EL1. Also controls exception catch on exception return to Non-secure EL1 in conjunction with EDECCR.NSR1.

Value	Meaning
0b0	If EDECCR.NSR1 is 0, then Exception Catch debug events are disabled for Non-secure EL1. If EDECCR.NSR1 is 1, then Exception Catch debug events are enabled for exception returns to Non-secure EL1.
0b1	If EDECCR.NSR1 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Non-secure EL1. If EDECCR.NSR1 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Non-secure EL1.

It is IMPLEMENTATION DEFINED whether a reset entry to an Exception level will generate an Exception Catch debug event.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0b0.

When Non-secure EL1 is implemented

NSE1, bit [5]

Coarse-grained exception catch for Non-secure EL1. Controls Exception Catch debug events for Non-secure EL1.

Value	Meaning
0b0	Exception Catch debug events are disabled for Non-secure EL1.

Value	Meaning
`0b1`	Exception Catch debug events are enabled for Non-secure EL1.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

NSE0, bit [4]

Access to this field is **RES0**.

Bit [3]

When FEAT_Debugv8p2 is implemented and EL3 is implemented

SE3, bit [3]

Controls exception catch on exception entry to EL3. Also controls exception catch on exception return to EL3 in conjunction with EDECCR.SR3.

Value	Meaning
`0b0`	If EDECCR.SR3 is 0, then Exception Catch debug events are disabled for EL3. If EDECCR.SR3 is 1, then Exception Catch debug events are enabled for exception returns to EL3.
`0b1`	If EDECCR.SR3 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to EL3. If EDECCR.SR3 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to EL3.

It is IMPLEMENTATION DEFINED whether a reset entry to an Exception level will generate an Exception Catch debug event.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

When FEAT_Debugv8p2 is not implemented and EL3 is implemented

SE3, bit [3]

Coarse-grained exception catch for EL3. Controls Exception Catch debug events for EL3.

Value	Meaning
`0b0`	Exception Catch debug events are disabled for EL3.
`0b1`	Exception Catch debug events are enabled for EL3.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

SE2, bit [2]

When FEAT_Debugv8p2 is implemented and FEAT_SEL2 is implemented:

Controls exception catch on exception entry to Secure EL2. Also controls exception catch on exception return to Secure EL2 in conjunction with EDECCR.SR2.

Value	Meaning
`0b0`	If EDECCR.SR2 is 0, then Exception Catch debug events are disabled for Secure EL2. If EDECCR.SR2 is 1, then Exception Catch debug events are enabled for exception returns to Secure EL2.
`0b1`	If EDECCR.SR2 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL2. If EDECCR.SR2 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL2.

It is IMPLEMENTATION DEFINED whether a reset entry to an Exception level will generate an Exception Catch debug event.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

Bit [1]

When FEAT_Debugv8p2 is implemented and Secure EL1 is implemented

SE1, bit [1]

Controls exception catch on exception entry to Secure EL1. Also controls exception catch on exception return to Secure EL1 in conjunction with EDECCR.SR1.

Value	Meaning
`0b0`	If EDECCR.SR1 is 0, then Exception Catch debug events are disabled for Secure EL1. If EDECCR.SR1 is 1, then Exception Catch debug events are enabled for exception returns to Secure EL1.

Value	Meaning
`0b1`	If EDECCR.SR1 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL1. If EDECCR.SR1 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL1.

It is IMPLEMENTATION DEFINED whether a reset entry to an Exception level will generate an Exception Catch debug event.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

When Secure EL1 is implemented

SE1, bit [1]

Coarse-grained exception catch for Secure EL1. Controls Exception Catch debug events for Secure EL1.

Value	Meaning
`0b0`	Exception Catch debug events are disabled for Secure EL1.
`0b1`	Exception Catch debug events are enabled for Secure EL1.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

SE0, bit [0]

Access to this field is **RES0**.

Accessing the EDECCR

EDECCR can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0x098	EDECCR

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and SoftwareLockStatus() access to this register is **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and !SoftwareLockStatus() access to this register is **RW**.
- Otherwise access to this register returns an ERROR.

A2.4.4 EDPRCR, External Debug Power/Reset Control Register

The EDPRCR characteristics are:

Purpose

Controls the PE functionality related to powerup, reset, and powerdown.

Attributes

EDPRCR is a 32-bit register.

Configuration

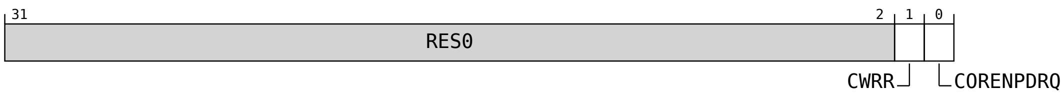
If FEAT_DoPD is implemented then all fields in this register are in the Core power domain.

CORENPDRQ is the only field that is mapped between the EDPRCR and DBGPRCR and DBGPRCR_EL1.

Field descriptions

The EDPRCR bit assignments are:

When FEAT_DoPD is implemented:



Bits [31:2]

Reserved, RES0.

Bit [1]

When FEAT_RME is implemented

CWRR, bit [1]

The PE ignores all writes to this bit.

Otherwise

CWRR, bit [1]

Warm reset request.

The extent of the reset is IMPLEMENTATION DEFINED, but must be one of:

- The request is ignored.
- Only this PE is Warm reset.
- This PE and other components of the system, possibly including other PEs, are Warm reset.

Arm deprecates use of this bit, and recommends that implementations ignore the request.

Value	Meaning
0b0	No action.
0b1	Request Warm reset.

This field is in the Core power domain

The PE ignores writes to this bit if any of the following are true:

- ExternalInvasiveDebugEnabled() == FALSE, EL3 is not implemented, and the implemented Security state is Non-secure state.
- ExternalSecureInvasiveDebugEnabled() == FALSE, EL3 is not implemented, and the implemented Security state is Secure state.
- ExternalSecureInvasiveDebugEnabled() == FALSE and EL3 is implemented.

In an implementation that includes the recommended external debug interface, this bit drives the DBGRSTREQ signal.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Accessing this field has the following behavior:

- RAZ/WI if any of the following are true:
  - OSLockStatus()
  - SoftwareLockStatus()
- Otherwise access is WO/RAZ

CORENPDRQ, bit [0]

Core no powerdown request. Requests emulation of powerdown.

This request is typically passed to an external power controller. This means that whether a request causes power up is dependent on the IMPLEMENTATION DEFINED nature of the system. The power controller must not allow the Core power domain to switch off while this bit is 1.

Value	Meaning
`0b0`	If the system responds to a powerdown request, it powers down Core power domain.
`0b1`	If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain.

When this bit reads as UNKNOWN, the PE ignores writes to this bit.

This field is in the Core power domain, and permitted accesses to this field map to the DBGPRCR.CORENPDRQ and DBGPRCR_EL1.CORENPDRQ fields.

In an implementation that includes the recommended external debug interface, this bit drives the DBGNOPWRDWN signal.

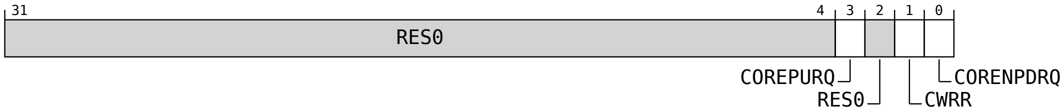
It is IMPLEMENTATION DEFINED whether this bit is reset to the Cold reset value on exit from an IMPLEMENTATION DEFINED software-visible retention state. For more information about retention states, see 'Core power domain power states'.

Writes to this bit are not prohibited by the IMPLEMENTATION DEFINED authentication interface. This means that a debugger can request emulation of powerdown regardless of whether invasive debug is permitted.

Accessing this field has the following behavior:

- When OSLockStatus(), access is UNKNOWN/WI
- When SoftwareLockStatus(), access is RO
- Otherwise access is RW

Otherwise:



Bits [31:4]

Reserved, RES0.

COREPURQ, bit [3]

Core powerup request. Allows a debugger to request that the power controller power up the core, enabling access to the debug register in the Core power domain, and that the power controller emulates powerdown.

This request is typically passed to an external power controller. This means that whether a request causes power up is dependent on the IMPLEMENTATION DEFINED nature of the system. The power controller must not allow the Core power domain to switch off while this bit is 1.

Value	Meaning
`0b0`	Do not request power up of the Core power domain.
`0b1`	Request power up of the Core power domain, and emulation of powerdown.

In an implementation that includes the recommended external debug interface, this bit drives the DBGPWRUPREQ signal.

This field is in the Debug power domain and can be read and written when the Core power domain is powered off.

Writes to this bit are not prohibited by the IMPLEMENTATION DEFINED authentication interface. This means that a debugger can request emulation of powerdown regardless of whether invasive debug is permitted.

The reset behavior of this field is:

- On a Debug reset, this field resets to `0b0`.

Accessing this field has the following behavior:

- When SoftwareLockStatus(), access is RO
- Otherwise access is RW

Bit [2]

Reserved, RES0.

Bit [1]

When FEAT_RME is implemented

CWRR, bit [1]

The PE ignores all writes to this bit.

Otherwise

CWRR, bit [1]

Warm reset request.

The extent of the reset is IMPLEMENTATION DEFINED, but must be one of:

- The request is ignored.
- Only this PE is Warm reset.
- This PE and other components of the system, possibly including other PEs, are Warm reset.

Arm deprecates use of this bit, and recommends that implementations ignore the request.

Value	Meaning
`0b0`	No action.
`0b1`	Request Warm reset.

This field is in the Core power domain

The PE ignores writes to this bit if any of the following are true:

- ExternalInvasiveDebugEnabled() == FALSE, EL3 is not implemented, and the implemented Security state is Non-secure state.
- ExternalSecureInvasiveDebugEnabled() == FALSE, EL3 is not implemented, and the implemented Security state is Secure state.
- ExternalSecureInvasiveDebugEnabled() == FALSE and EL3 is implemented.

In an implementation that includes the recommended external debug interface, this bit drives the DBGRSTREQ signal.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0b0`.

Accessing this field has the following behavior:

- RAZ/WI if any of the following are true:
  - !IsCorePowered()
  - DoubleLockStatus()
  - OSLockStatus()
  - SoftwareLockStatus()
- Otherwise access is WO/RAZ

CORENPDRQ, bit [0]

Core no powerdown request. Requests emulation of powerdown.

This request is typically passed to an external power controller. This means that whether a request causes power up is dependent on the IMPLEMENTATION DEFINED nature of the system. The power controller must not allow the Core power domain to switch off while this bit is 1.

Value	Meaning
`0b0`	If the system responds to a powerdown request, it powers down Core power domain.
`0b1`	If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain.

When this bit reads as UNKNOWN, the PE ignores writes to this bit.

This field is in the Core power domain, and permitted accesses to this field map to the DBGPRCR.CORENPDRQ and DBGPRCR_EL1.CORENPDRQ fields.

In an implementation that includes the recommended external debug interface, this bit drives the DBGNOPWRDWN signal.

It is IMPLEMENTATION DEFINED whether this bit is reset to the value of EDPRCR.COREPURQ on exit from an IMPLEMENTATION DEFINED software-visible retention state. For more information about retention states, see 'Core power domain power states'.

Writes to this bit are not prohibited by the IMPLEMENTATION DEFINED authentication interface. This means that a debugger can request emulation of powerdown regardless of whether invasive debug is permitted.

The reset behavior of this field is:

- On a Cold reset, this field resets to the value in EDPRCR.COREPURQ.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - !IsCorePowered()
  - DoubleLockStatus()
  - OSLockStatus()
- When SoftwareLockStatus(), access is RO
- Otherwise access is RW

Accessing the EDPRCR

On permitted accesses to the register, other access controls affect the behavior of some fields. See the field descriptions for more information.

EDPRCR can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0x310	EDPRCR

This interface is accessible as follows:

- When (FEAT_DoPD is not implemented or IsCorePowered()) and SoftwareLockStatus() access to this register is **RO**.
- When (FEAT_DoPD is not implemented or IsCorePowered()) and !SoftwareLockStatus() access to this register is **RW**.
- Otherwise access to this register returns an ERROR.

A2.4.5 EDPRSR, External Debug Processor Status Register

The EDPRSR characteristics are:

Purpose

Holds information about the reset and powerdown state of the PE.

Attributes

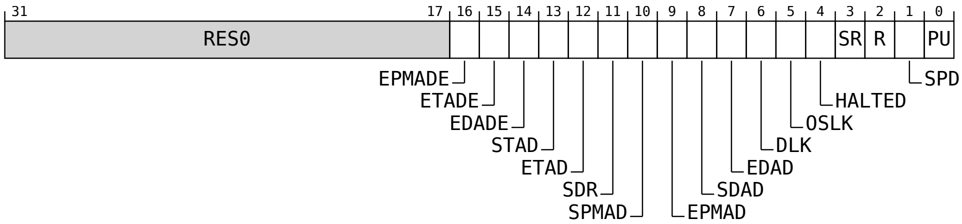
EDPRSR is a 32-bit register.

Configuration

If FEAT_DoPD is implemented then all fields in this register are in the Core power domain.

Field descriptions

The EDPRSR bit assignments are:



Bits [31:17]

Reserved, RES0.

EPMADE, bit [16]

When FEAT_RME is implemented and FEAT_PMUv3 is implemented:

External Performance Monitors Access Disable Extended Status. Together with EDPRSR.EPMAD, reports whether access to Performance Monitor registers by an external debugger is permitted.

For a description of the values derived by evaluating EPMAD and EPMADE together, see EDPRSR.EPMAD.

Otherwise:

RES0

ETADE, bit [15]

When FEAT_RME is implemented, external debugger access to the PE Trace Unit registers is implemented and FEAT_TRBE is implemented:

External Trace Access Disable Extended Status. Together with EDPRSR.ETAD, reports whether access to PE Trace Unit registers by an external debugger is permitted.

For a description of the values derived by evaluating ETAD and ETADE together, see EDPRSR.ETAD.

Otherwise:

RES0

EDADE, bit [14]

When FEAT_RME is implemented:

External Debug Access Disable Extended Status. Together with EDPRSR.EDAD, reports whether access to breakpoint registers, watchpoint registers, and OSLAR_EL1 by an external debugger is permitted.

For a description of the values derived by evaluating EDAD and EDADE together, see EDPRSR.EDAD.

Otherwise:

RES0

STAD, bit [13]

When external debugger access to the PE Trace Unit registers is implemented and FEAT_TRBE is implemented:

Sticky ETAD error. Set to 1 when a Non-secure external debug interface access to an external trace register returns an error because AllowExternalTraceAccess() == FALSE for the access.

Value	Meaning
0b0	Since EDPRSR was last read, no external accesses to the PE Trace Unit registers have failed because AllowExternalTraceAccess() was FALSE for the access.
0b1	Since EDPRSR was last read, at least one external access to the PE Trace Unit registers has failed because AllowExternalTraceAccess() was FALSE for the access.

If IsCorePowered() == TRUE, the Core power domain is powered up, then, following a read of EDPRSR:

- If FEAT_DoubleLock is not implemented or DoubleLockStatus() == FALSE then this bit clears to 0.
- If FEAT_DoubleLock is implemented and DoubleLockStatus() == TRUE then it is CONSTRAINED UNPREDICTABLE whether this bit clears to 0 or is unchanged.

This bit is in the Core power domain.

If FEAT_DoPD is implemented, FEAT_DoubleLock is not implemented.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0b0.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - DoubleLockStatus()
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - EDPRSR.R == '1'
- Otherwise access is RC/WI

Otherwise:

RES0

Bit [12]

When FEAT_RME is implemented, external debugger access to the PE Trace Unit registers is implemented and FEAT_TRBE is implemented

ETAD, bit [12]

External Trace Access Disable Status. Together with EDPRSR.ETADE, reports whether access to PE Trace Unit registers by an external debugger is permitted.

ETADE	ETAD	Meaning
0b0	0b0	Access to PE Trace Unit registers by an external debugger is permitted.
0b0	0b1	Root and Secure access to PE Trace Unit registers by an external debugger is permitted. Realm and Non-secure access to PE Trace Unit registers by an external debugger is not permitted.
0b1	0b0	Root and Realm access to PE Trace Unit registers by an external debugger is permitted. Secure and Non-secure access to PE Trace Unit registers by an external debugger is not permitted.
0b1	0b1	Root access to PE Trace Unit registers by an external debugger is permitted. Secure, Non-secure, and Realm access to PE Trace Unit registers by an external debugger is not permitted.

When external debugger access to the PE Trace Unit registers is implemented and FEAT_TRBE is implemented

ETAD, bit [12]

External Trace Access Disable status.

Value	Meaning
`0b0`	External Non-secure PE Trace Unit accesses enabled. AllowExternalTraceAccess() == TRUE for a Non-secure access.
`0b1`	External Non-secure PE Trace Unit accesses disabled. AllowExternalTraceAccess() == FALSE for a Non-secure access.

This bit is in the Core power domain.

If FEAT_DoPD is implemented, FEAT_DoubleLock is not implemented.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - DoubleLockStatus()
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - EDPRSR.R == '1'
- Otherwise access is RO

Otherwise:

RES0

SDR, bit [11]

Sticky Debug Restart. Set to 1 when the PE exits Debug state.

Permitted values are:

Value	Meaning
`0b0`	The PE has not restarted since EDPRSR was last read.
`0b1`	The PE has restarted since EDPRSR was last read.

If a reset occurs when the PE is in Debug state, the PE exits Debug state. SDR is UNKNOWN on Warm reset, meaning a debugger must also use the SR bit to determine whether the PE has left Debug state.

If The Core power domain is powered up, then following a read of EDPRSR:

- If FEAT_DoubleLock is not implemented or DoubleLockStatus() == FALSE this bit clears to 0.
- If FEAT_DoubleLock is implemented and DoubleLockStatus() == TRUE, it is CONSTRAINED UNPREDICTABLE whether this bit clears to 0 or is unchanged.

This field is in the Core power domain.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - DoubleLockStatus()
  - EDPRSR.R == '1'
- When SoftwareLockStatus(), access is RO
- Otherwise access is RC/WI

Bit [10]

When FEAT_Debugv8p4 is implemented

SPMAD, bit [10]

Sticky EPMAD error. Set to 1 if an external debug interface access to a Performance Monitors register returns an error because AllowExternalPMUAccess() == FALSE.

Permitted values are:

Value	Meaning
`0b0`	No Non-secure external debug interface accesses to the external Performance Monitors registers have failed because AllowExternalPMUAccess() == FALSE for the access since EDPRSR was last read.
`0b1`	At least one Non-secure external debug interface access to the external Performance Monitors register has failed and returned an error because AllowExternalPMUAccess() == FALSE for the access since EDPRSR was last read.

If the Core power domain is powered up, then, following a read of EDPRSR:

- If FEAT_DoubleLock is not implemented or DoubleLockStatus() == FALSE, this bit clears to 0.
- If FEAT_DoubleLock is implemented, and DoubleLockStatus() == TRUE, it is CONSTRAINED UNPREDICTABLE whether this bit clears to 0 or is unchanged.

This field is in the Core power domain.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:

- (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
- DoubleLockStatus()
- EDPRSR.R == '1'
- When SoftwareLockStatus(), access is RO
- Otherwise access is RC/WI

Otherwise

SPMAD, bit [10]

Sticky EPMAD error.

Value	Meaning
`0b0`	No external debug interface accesses to the Performance Monitors registers have failed because AllowExternalPMUAccess() == FALSE since EDPRSR was last read.
`0b1`	At least one external debug interface access to the Performance Monitors registers has failed and returned an error because AllowExternalPMUAccess() == FALSE since EDPRSR was last read.

If the Core power domain is powered up, then, following a read of EDPRSR:

- If FEAT_DoubleLock is not implemented or DoubleLockStatus() == FALSE, this bit clears to 0.
- If FEAT_DoubleLock is implemented, and DoubleLockStatus() == TRUE, it is CONSTRAINED UNPREDICTABLE whether this bit clears to 0 or is unchanged.

This field is in the Core power domain.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - OSLockStatus()
  - DoubleLockStatus()
  - EDPRSR.R == '1'
- When SoftwareLockStatus(), access is RO
- Otherwise access is RC/WI

Bit [9]

When FEAT_RME is implemented and FEAT_PMUv3 is implemented

EPMAD, bit [9]

External Performance Monitors Access Disable Status. Together with EDPRSR.EPMADE, reports whether access to Performance Monitor registers by an external debugger is permitted.

EPMADE	EPMAD	Meaning
0b0	0b0	Access to Performance Monitor registers by an external debugger is permitted.

EPMADE	EPMAD	Meaning
0b0	0b1	Root and Secure access to Performance Monitor registers by an external debugger is permitted. Realm and Non-secure access to Performance Monitor registers by an external debugger is not permitted.
0b1	0b0	Root and Realm access to Performance Monitor registers by an external debugger is permitted. Secure and Non-secure access to Performance Monitor registers by an external debugger is not permitted.
0b1	0b1	Root access to Performance Monitor registers by an external debugger is permitted. Secure, Non-secure, and Realm access to Performance Monitor registers by an external debugger is not permitted.

When FEAT_Debugv8p4 is implemented and FEAT_PMUv3 is implemented

EPMAD, bit [9]

External Performance Monitors Non-secure Access Disable status.

Value	Meaning
`0b0`	External Non-secure Performance Monitors access enabled. AllowExternalPMUAccess() == TRUE for a Non-secure access.
`0b1`	External Non-secure Performance Monitors access disabled. AllowExternalPMUAccess() == FALSE for a Non-secure access.

This field is in the Core power domain.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - DoubleLockStatus()
  - EDPRSR.R == '1'
- Otherwise access is RO

When FEAT_PMUv3 is implemented

EPMAD, bit [9]

External Performance Monitors access disable status.

Value	Meaning
`0b0`	External Performance Monitors access enabled. AllowExternalPMUAccess() == TRUE.
`0b1`	External Performance Monitors access disabled. AllowExternalPMUAccess() == FALSE.

This field is in the Core power domain.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:

- (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
- OSLockStatus()
- DoubleLockStatus()
- EDPRSR.R == '1'

- Otherwise access is RO

Otherwise:

RES0

Bit [8]

When FEAT_Debugv8p4 is implemented

SDAD, bit [8]

Sticky EDAD error. Set to 1 if an external debug interface access to a debug register returns an error because AllowExternalDebugAccess() == FALSE.

Value	Meaning
`0b0`	No Non-secure external debug interface accesses to the debug registers have failed because AllowExternalDebugAccess() == FALSE for the access since EDPRSR was last read.
`0b1`	At least one Non-secure external debug interface access to the debug registers has failed and returned an error because AllowExternalDebugAccess() == FALSE for the access since EDPRSR was last read.

If the Core power domain is powered up, then, following a read of EDPRSR:

- If FEAT_DoubleLock is not implemented or DoubleLockStatus() == FALSE this bit clears to 0.
- If FEAT_DoubleLock is implemented and DoubleLockStatus() == TRUE, it is CONSTRAINED UNPREDICTABLE whether this bit clears to 0 or is unchanged.

This field is in the Core power domain.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - DoubleLockStatus()
  - EDPRSR.R == '1'
- Otherwise access is RO

Otherwise

SDAD, bit [8]

Sticky EDAD error. Set to 1 if an external debug interface access to a debug register returns an error because AllowExternalDebugAccess() == FALSE.

Value	Meaning
`0b0`	No external debug interface accesses to the debug registers have failed because AllowExternalDebugAccess() == FALSE since EDPRSR was last read.
`0b1`	At least one external debug interface access to the debug registers has failed and returned an error because AllowExternalDebugAccess() == FALSE since EDPRSR was last read.

If the Core power domain is powered up, then, following a read of EDPRSR:

- If FEAT_DoubleLock is not implemented or DoubleLockStatus() == FALSE this bit clears to 0.
- If FEAT_DoubleLock is implemented and DoubleLockStatus() == TRUE, it is CONSTRAINED UNPREDICTABLE whether this bit clears to 0 or is unchanged.

This bit is UNKNOWN on reads if OSLockStatus() == TRUE and external debug writes to OSLAR_EL1 do not return an error when AllowExternalDebugAccess() == FALSE.

This field is in the Core power domain.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - DoubleLockStatus()
  - EDPRSR.R == '1'
- Otherwise access is RO

Bit [7]

When FEAT_RME is implemented

EDAD, bit [7]

External Debug Access Disable Status. Together with EDPRSR.EDADE, reports whether access to breakpoint registers, watchpoint registers, and OSLAR_EL1 by an external debugger is permitted.

EDADE	EDAD	Meaning
0b0	0b0	Access to Debug registers by an external debugger is permitted.
0b0	0b1	Root and Secure access to Debug registers by an external debugger is permitted. Realm and Non-secure access to Debug registers by an external debugger is not permitted.
0b1	0b0	Root and Realm access to Debug registers by an external debugger is permitted. Secure and Non-secure access to Debug registers by an external debugger is not permitted.
0b1	0b1	Root access to Debug registers by an external debugger is permitted. Secure, Non-secure, and Realm access to Debug registers by an external debugger is not permitted.

When FEAT_Debugv8p4 is implemented

EDAD, bit [7]

External Debug Access Disable status.

Value	Meaning
0b0	External Non-secure access to breakpoint registers, watchpoint registers, and OSLAR_EL1 enabled. AllowExternalDebugAccess() == TRUE for a Non-secure access.
0b1	External Non-secure access to breakpoint registers, watchpoint registers, and OSLAR_EL1 disabled. AllowExternalDebugAccess() == FALSE for a Non-secure access.

This field is in the Core power domain.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - DoubleLockStatus()
  - EDPRSR.R == '1'
- Otherwise access is RO

When FEAT_Debugv8p2 is implemented

EDAD, bit [7]

External Debug Access Disable status.

Value	Meaning
0b0	External access to breakpoint registers, watchpoint registers, and OSLAR_EL1 enabled. AllowExternalDebugAccess() == TRUE.
0b1	External access to breakpoint registers, watchpoint registers, and OSLAR_EL1 disabled. AllowExternalDebugAccess() == FALSE.

This bit is not valid and reads UNKNOWN if OSLockStatus() == TRUE and external debug writes to OSLAR_EL1 do not return an error when AllowExternalDebugAccess() == FALSE.

This field is in the Core power domain.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - DoubleLockStatus()
  - EDPRSR.R == '1'
- Otherwise access is RO

Otherwise

EDAD, bit [7]

External Debug Access Disable status.

Value	Meaning
0b0	External access to breakpoint registers, watchpoint registers, and OSLAR_EL1 enabled. AllowExternalDebugAccess() == TRUE.
0b1	External access to breakpoint registers, watchpoint registers disabled. It is IMPLEMENTATION DEFINED whether accesses to OSLAR_EL1 are enabled or disabled. AllowExternalDebugAccess() == FALSE.

This field is in the Core power domain.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - DoubleLockStatus()
  - EDPRSR.R == '1'
- Otherwise access is RO

Bit [6]

When FEAT_Debugv8p4 is implemented

DLK, bit [6]

This field is RES0.

When FEAT_Debugv8p2 is implemented and FEAT_DoubleLock is implemented

DLK, bit [6]

Double Lock.

From Armv8.2, this field is deprecated.

This field is in the Core power domain.

Accessing this field has the following behavior:

- RAZ/WI if all of the following are true:
  - IsCorePowered()
  - !DoubleLockStatus()
- Otherwise access is UNKNOWN/WI

When FEAT_DoubleLock is implemented

DLK, bit [6]

Double Lock.

This field returns the result of the pseudocode function DoubleLockStatus().

If the Core power domain is powered up and DoubleLockStatus() == TRUE, it is IMPLEMENTATION DEFINED whether:

- EDPRSR.PU reads as 1, EDPRSR.DLK reads as 1, and EDPRSR.SPD is UNKNOWN.
- EDPRSR.PU reads as 0, EDPRSR.DLK is UNKNOWN, and EDPRSR.SPD reads as 0.

This field is in the Core power domain.

Value	Meaning
0b0	DoubleLockStatus() returns FALSE.
0b1	DoubleLockStatus() returns TRUE and the Core power domain is powered up.

Accessing this field has the following behavior:

- UNKNOWN/WI if all of the following are true:
  - !IsFeatureImplemented(FEAT_DoPD)
  - !IsCorePowered()
- Otherwise access is RO

Otherwise:

RES0

OSLK, bit [5]

OS Lock status bit.

A read of this bit returns the value of OSLSR_EL1.OSLK.

This field is in the Core power domain.

Accessing this field has the following behavior:

- UNKNOWN/WI if all of the following are true:
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - DoubleLockStatus()
  - EDPRSR.R == ‘1’
- Otherwise access is RO

HALTED, bit [4]

Halted status bit.

Value	Meaning
0b0	PE is in Non-debug state.
0b1	PE is in Debug state.

This field is in the Core power domain.

Accessing this field has the following behavior:

- UNKNOWN/WI if all of the following are true:
  - !IsFeatureImplemented(FEAT_DoPD)
  - !IsCorePowered()

- Otherwise access is RO

SR, bit [3]

Sticky core Reset status bit.

Permitted values are:

Value	Meaning
0b0	The non-debug logic of the PE is not in reset state and has not been reset since the last time EDPRSR was read.
0b1	The non-debug logic of the PE is in reset state or has been reset since the last time EDPRSR was read.

If EDPRSR.PU reads as 1 and EDPRSR.R reads as 0, which means that the Core power domain is in a powerup state and that the non-debug logic of the PE is not in reset state, then following a read of EDPRSR:

- If FEAT_DoubleLock is not implemented or DoubleLockStatus() == FALSE this bit clears to 0.
- If FEAT_DoubleLock is implemented and DoubleLockStatus() == TRUE, it is UNPREDICTABLE whether this bit clears to 0 or is unchanged.

This field is in the Core power domain.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0b1.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())
  - DoubleLockStatus()
- When SoftwareLockStatus(), access is RO
- Otherwise access is RC/WI

R, bit [2]

PE Reset status bit.

Permitted values are:

Value	Meaning
0b0	The non-debug logic of the PE is not in reset state.
0b1	The non-debug logic of the PE is in reset state.

If FEAT_DoubleLock is implemented, the PE is in reset state, and the PE entered reset state with the OS Double Lock locked this bit has a CONSTRAINED UNPREDICTABLE value. For more information, see 'EDPRSR.{DLK, R} and reset state'.

This field is in the Core power domain.

Accessing this field has the following behavior:

- UNKNOWN/WI if any of the following are true:
  - (!IsFeatureImplemented(FEAT_DoPD) and !IsCorePowered())

  - DoubleLockStatus()
- Otherwise access is RO

SPD, bit [1]

Sticky core Powerdown status bit.

If FEAT_DoubleLock is implemented and DoubleLockStatus() == TRUE, then:

- If FEAT_Debugv8p2 is implemented, this bit reads as 0.
- If FEAT_Debugv8p2 is not implemented, this bit might read as 0 or 1.

For more information, see 'EDPRSR.{DLK, SPD, PU} and the Core power domain'.

Value	Meaning
`0b0`	If EDPRSR.PU is 0, it is not known whether the state of the debug registers in the Core power domain is lost. If EDPRSR.PU is 1, the state of the debug registers in the Core power domain has not been lost.
`0b1`	The state of the debug registers in the Core power domain has been lost.

If the Core power domain is powered up, then, following a read of EDPRSR:

- If FEAT_DoubleLock is not implemented or DoubleLockStatus() == FALSE this bit clears to 0.
- If FEAT_DoubleLock is implemented and DoubleLockStatus() == TRUE, it is CONSTRAINED UNPREDICTABLE whether this bit clears to 0 or is unchanged.

When FEAT_DoPD is not implemented and the Core power domain is in either retention or powerdown state, the value of EDPRSR.SPD is IMPLEMENTATION DEFINED. For more information, see 'EDPRSR.SPD when the Core domain is in either retention or powerdown state'.

EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see 'EDPRSR.{DLK, SPD, PU} and the Core power domain'.

This field is in the Core power domain.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b1`.

Accessing this field has the following behavior:

- RAZ/WI if all of the following are true:
  - !IsFeatureImplemented(FEAT_DoPD)
  - !IsCorePowered()
- UNKNOWN/WI if all of the following are true:
  - IsCorePowered()
  - DoubleLockStatus()
- Otherwise access is RO

Bit [0]

When FEAT_DoPD is implemented

PU, bit [0]

Core powerup status bit.

Access to this field is **RAO/WI**.

When FEAT_Debugv8p2 is implemented

PU, bit [0]

Core Powerup status bit. Indicates whether the debug registers in the Core power domain can be accessed.

Value	Meaning
`0b0`	Either the Core power domain is in a low-power or powerdown state, or FEAT_DoubleLock is implemented and DoubleLockStatus() == TRUE, meaning the debug registers in the Core power domain cannot be accessed.
`0b1`	The Core power domain is in a powerup state, and either FEAT_DoubleLock is not implemented or DoubleLockStatus() == FALSE, meaning the debug registers in the Core power domain can be accessed.

If FEAT_DoubleLock is implemented, the PE is in reset state, and the PE entered reset state with the OS Double Lock locked this bit has a CONSTRAINED UNPREDICTABLE value. For more information, see 'EDPRSR.{DLK, R} and reset state'

EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see 'EDPRSR.{DLK, SPD, PU} and the Core power domain'

Access to this field is **RO**.

Otherwise

PU, bit [0]

Core Powerup status bit. Indicates whether the debug registers in the Core power domain can be accessed.

When the Core power domain is powered-up and DoubleLockStatus() == TRUE, then the value of EDPRSR.PU is IMPLEMENTATION DEFINED. See the description of the DLK bit for more information.

Otherwise, permitted values are:

Value	Meaning
`0b0`	Core power domain is in a low-power or powerdown state where the debug registers in the Core power domain cannot be accessed.
`0b1`	Core power domain is in a powerup state where the debug registers in the Core power domain can be accessed.

If FEAT_DoubleLock is implemented, the Core power domain is powered up, and DoubleLockStatus() == TRUE, it is IMPLEMENTATION DEFINED whether this bit reads as 0 or 1.

If FEAT_DoubleLock is implemented, the PE is in reset state, and the PE entered reset state with the OS Double Lock locked this bit has a CONSTRAINED UNPREDICTABLE value. For more information see 'EDPRSR.{DLK, R} and reset state'

EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see 'EDPRSR.{DLK, SPD, PU} and the Core power domain'.

Access to this field is **RO**.

Accessing the EDPRSR

On permitted accesses to the register, other access controls affect the behavior of some fields. See the field descriptions for more information.

If the Core power domain is powered up (EDPRSR.PU == 1), then following a read of EDPRSR:

- If FEAT_DoubleLock is not implemented or DoubleLockStatus() == FALSE, then:
  - EDPRSR.{SDR, SPMAD, SDAD, SPD} are cleared to 0.
  - EDPRSR.SR is cleared to 0 if the non-debug logic of the PE is not in reset state (EDPRSR.R == 0).
- If FEAT_DoubleLock is implemented and DoubleLockStatus() == TRUE, it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs.

If FEAT_DoPD is not implemented and the Core power domain is powered down (EDPRSR.PU == 0), then:

- EDPRSR.{SDR, SPMAD, SDAD, SR} are all UNKNOWN, and are either reset or restored on being powered up.
- EDPRSR.SPD is not cleared following a read of EDPRSR. See the SPD bit description for more information.

The clearing of bits is an indirect write to EDPRSR.

EDPRSR can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0x314	EDPRSR

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered() access to this register is **RO**.
- Otherwise access to this register returns an ERROR.

A2.4.6 EDSCR, External Debug Status and Control Register

The EDSCR characteristics are:

Purpose

Main control register for the debug implementation.

Attributes

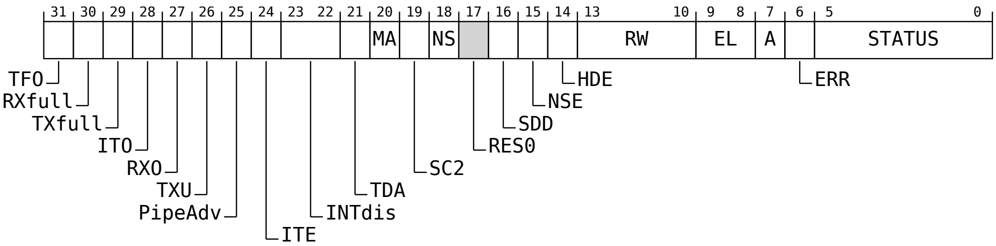
EDSCR is a 32-bit register.

Configuration

External register EDSCR bits [30:29] are architecturally mapped to AArch64 system register MDCCSR_EL0[30:29].

Field descriptions

The EDSCR bit assignments are:



TFO, bit [31]

When FEAT_TRF is implemented:

Trace Filter Override. Overrides the Trace Filter controls allowing the external debugger to trace any visible Exception level.

Value	Meaning
`0b0`	Trace Filter controls are not affected.
`0b1`	Trace Filter controls in TRFCR_EL1 and TRFCR_EL2 are ignored. Trace Filter controls TRFCR and HTRFCR are ignored.

When OSLSR_EL1.OSLK is 1, this bit can be indirectly read and written through the MDSCR_EL1 and DBGDSCRext System registers.

This bit is ignored by the PE when any of the following is true:

- ExternalSecureNoninvasiveDebugEnabled() is FALSE and the Effective value of MDCR_EL3.STE is 1.
- FEAT_RME is implemented, ExternalRealmNoninvasiveDebugEnabled() is FALSE, and the Effective value of MDCR_EL3.RLTE is 1.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

RXfull, bit [30]

DTRRX full.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Access to this field is **RO**.

TXfull, bit [29]

DTRTX full.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Access to this field is **RO**.

ITO, bit [28]

ITR overrun. Set to 0 on entry to Debug state.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access is UNKNOWN/WI
- Otherwise access is RO

RXO, bit [27]

DTRRX overrun.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Access to this field is **RO**.

TXU, bit [26]

DTRTX underrun.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Access to this field is **RO**.

PipeAdv, bit [25]

Pipeline Advance. Indicates that software execution is progressing.

Value	Meaning
`0b0`	No progress has been made by the PE since the last time this field was cleared to zero by writing 1 to EDRCR.CSPA.
`0b1`	Progress has been made by the PE since the last time this field was cleared to zero by writing 1 to EDRCR.CSPA.

The architecture does not define precisely when this field is set to 1. It requires only that this happen periodically in Non-debug state to indicate that software execution is progressing. For example, a PE might set this field to 1 each time the PE retires one or more instructions, or at periodic intervals during the progression of an instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Access to this field is **RO**.

ITE, bit [24]

ITR empty.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access is UNKNOWN/WI
- Otherwise access is RO

Bits [23:22]

When FEAT_RME is implemented

INTdis, bits [23:22]

Interrupt disable. Disables taking interrupts in Non-debug state.

Value	Meaning
0b00	This bit has no effect on the masking of interrupts.
0b01	If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure state are masked. If ExternalSecureInvasiveDebugEnabled() is TRUE, then all interrupts taken to Secure state are masked. If ExternalRootInvasiveDebugEnabled() is TRUE, then all interrupts taken to Root state are masked. If ExternalRealmInvasiveDebugEnabled() is TRUE, then all interrupts taken to Realm state are masked.

All interrupts includes virtual and SError interrupts.

When OSLSR_EL1.OSLK is 1, this field can be indirectly read and written through the MDSCR_EL1 and DBGDSCRext System registers.

The Effective value of this field is 0b00 when ExternalInvasiveDebugEnabled() is FALSE.

When FEAT_RME is implemented, bit[23] of this register is RES0.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0b00.

When FEAT_Debugv8p4 is implemented

INTdis, bits [23:22]

Interrupt disable. Disables taking interrupts in Non-debug state.

Value	Meaning
0b00	Masking of interrupts is controlled by PSTATE and interrupt routing controls.
0b01	If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure state are masked. If ExternalSecureInvasiveDebugEnabled() is TRUE, then all interrupts taken to Secure state are masked.

All interrupts includes virtual and SError interrupts.

When OSLSR_EL1.OSLK is 1, this field can be indirectly read and written through the MDSCR_EL1 and DBGDSCRext System registers.

The Effective value of this field is 0b00 when ExternalInvasiveDebugEnabled() is FALSE.

When FEAT_Debugv8p4 is implemented, bit[23] of this register is RES0.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0b00.

Otherwise

INTdis, bits [23:22]

Interrupt disable. Disables taking interrupts in Non-debug state.

Value	Meaning
0b00	Masking of interrupts is controlled by PSTATE and interrupt routing controls.
0b01	If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure EL1 are masked.
0b10	If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure state are masked. If ExternalSecureInvasiveDebugEnabled() is TRUE, then all interrupts taken to Secure EL1 are masked.
0b11	If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure state are masked. If ExternalSecureInvasiveDebugEnabled() is TRUE, then all interrupts taken to Secure state are masked.

All interrupts includes virtual and SError interrupts.

When OSLSR_EL1.OSLK is 1, this field can be indirectly read and written through the MDSCR_EL1 and DBGDSCRext System registers.

The Effective value of this field is 0b00 when ExternalInvasiveDebugEnabled() is FALSE.

Support for the values 0b01 and 0b10 is IMPLEMENTATION DEFINED. If these values are not supported, they are reserved. If programmed with a reserved value, the PE behaves as if INTdis has been programmed with a defined value, other than for a direct read of EDSCR, and the value returned by a read of EDSCR.INTdis is UNKNOWN.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b00`.

TDA, bit [21]

Traps accesses to the following debug System registers:

- AArch64: DBGBCR<n>_EL1, DBGBVR<n>_EL1, DBGWCR<n>_EL1, DBGWVR<n>_EL1.
- AArch32: DBGBCR<n>, DBGBVR<n>, DBGBXVR<n>, DBGWCR<n>, DBGWVR<n>.

Value	Meaning
`0b0`	Accesses to debug System registers do not generate a Software Access Debug event.
`0b1`	Accesses to debug System registers generate a Software Access Debug event, if OSLSR_EL1.OSLK is 0 and if halting is allowed.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

MA, bit [20]

Memory access mode. Controls the use of memory-access mode for accessing ITR and the DCC. This bit is ignored if in Non-debug state and set to zero on entry to Debug state.

Possible values of this field are:

Value	Meaning
`0b0`	Normal access mode.
`0b1`	Memory access mode.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

SC2, bit [19]

When FEAT_PCSRv8 is implemented, (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented) and FEAT_PCSRv8p2 is not implemented:

Sample CONTEXTIDR_EL2. Controls whether the PC Sample-based Profiling Extension samples CONTEXTIDR_EL2 or VTTBR_EL2.VMID.

Value	Meaning
`0b0`	Sample VTTBR_EL2.VMID.
`0b1`	Sample CONTEXTIDR_EL2.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Otherwise:

RES0

Bit [18]

When FEAT_RME is implemented

NS, bit [18]

Non-secure status. Together with the NSE field, gives the current Security state:

NSE	NS	Meaning
0b0	0b0	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access is UNKNOWN/WI
- Otherwise access is RO

Otherwise

NS, bit [18]

Non-secure status. In Debug state, gives the current Security state:

Value	**Meaning**
`0b0`	Secure state.
`0b1`	Non-secure state.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access is UNKNOWN/WI
- Otherwise access is RO

Bit [17]

Reserved, RES0.

Bit [16]

When FEAT_RME is implemented

SDD, bit [16]

Secure debug disabled.

Reports the inverse of ExternalRootInvasiveDebugEnabled().

Access to this field is **RO**.

Otherwise

SDD, bit [16]

Secure debug disabled.

On entry to Debug state:

- If entering in Secure state, SDD is set to 0.
- If entering in Non-secure state, SDD is set to the inverse of ExternalSecureInvasiveDebugEnabled().

In Debug state, the value of the SDD bit does not change, even if ExternalSecureInvasiveDebugEnabled() changes.

In Non-debug state:

- SDD returns the inverse of ExternalSecureInvasiveDebugEnabled(). If the authentication signals that control ExternalSecureInvasiveDebugEnabled() change, a context synchronization event is required to guarantee their effect.
- This bit is unaffected by the Security state of the PE.

If EL3 is not implemented and the implementation is Non-secure, this bit is RES1.

Access to this field is **RO**.

NSE, bit [15]

When FEAT_RME is implemented:

Together with the NS field, this field gives the current Security state.

For a description of the values derived by evaluating NS and NSE together, see EDSCR.NS.

In Non-debug state, this bit is UNKNOWN.

Access to this field is **RO**.

Otherwise:

RES0

HDE, bit [14]

Halting debug enable.

Value	Meaning
`0b0`	Halting disabled for Breakpoint, Watchpoint and Halt Instruction debug events.
`0b1`	Halting enabled for Breakpoint, Watchpoint and Halt Instruction debug events.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

RW, bits [13:10]

Exception level Execution state status. In Debug state, each bit gives the current Execution state of each Exception level.

Value	Meaning	Applies
`0b1111`	Any of the following: • The PE is in Non-debug state. • The PE is at EL0 using AArch64. • The PE is not at EL0, and EL1, EL2, and EL3 are using AArch64.	
`0b1110`	The PE is in Debug state at EL0. EL0 is using AArch32. EL1, EL2, and EL3 are using AArch64.	When AArch32 is supported
`0b110x`	The PE is in Debug state. EL0 and EL1 are using AArch32. EL2 is enabled in the current Security state and is using AArch64. If implemented, EL3 is using AArch64.	When AArch32 is supported and EL2 is implemented
`0b10xx`	The PE is in Debug state. EL0 and EL1 are using AArch32. EL2 is not implemented, disabled in the current Security state, or using AArch32. EL3 is using AArch64.	When AArch32 is supported and EL3 is implemented
`0b0xxx`	The PE is in Debug state. All Exception levels are using AArch32.	When AArch32 is supported

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access is RAO/WI
- Otherwise access is RO

EL, bits [9:8]

Exception level. In Debug state, gives the current Exception level of the PE.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access is RAZ/WI
- Otherwise access is RO

A, bit [7]

SError interrupt pending. In Debug state, indicates whether an SError interrupt is pending:

- If HCR_EL2.{AMO, TGE} = {1, 0}, EL2 is enabled in the current Security state, and the PE is executing at EL0 or EL1, a virtual SError interrupt.
- Otherwise, a physical SError interrupt.

Value	Meaning
`0b0`	No SError interrupt pending.
`0b1`	SError interrupt pending.

A debugger can read EDSCR to check whether an SError interrupt is pending without having to execute further instructions. A pending SError might indicate data from target memory is corrupted.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access is UNKNOWN/WI
- Otherwise access is RO

ERR, bit [6]

Cumulative error flag. This bit is set to 1 following exceptions in Debug state and on any signaled overrun or underrun on the DTR or EDITR.

The reset behavior of this field is:

- On a Cold reset, this field resets to `0b0`.

Access to this field is **RO**.

STATUS, bits [5:0]

Debug status flags.

Value	Meaning
`0b000001`	PE is restarting, exiting Debug state.
`0b000010`	PE is in Non-debug state.
`0b000111`	Breakpoint.
`0b010011`	External debug request.
`0b011011`	Halting step, normal.
`0b011111`	Halting step, exclusive.
`0b100011`	OS Unlock Catch.
`0b100111`	Reset Catch.
`0b101011`	Watchpoint.
`0b101111`	HLT instruction.
`0b110011`	Software access to debug register.
`0b110111`	Exception Catch.
`0b111011`	Halting step, no syndrome.

All other values of STATUS are reserved.

Access to this field is **RO**.

Accessing the EDSCR

EDSCR can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0x088	EDSCR

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and SoftwareLockStatus() access to this register is **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and !SoftwareLockStatus() access to this register is **RW**.
- Otherwise access to this register returns an ERROR.

A2.4.7 ERR<n>ADDR, Error Record Address Register, n = 0 - 65534

The ERR<n>ADDR characteristics are:

Purpose

If an address is associated with a detected error, then it is written to ERR<n>ADDR when the error is recorded. It is IMPLEMENTATION DEFINED how the recorded address maps to the software-visible physical address. Software might have to reconstruct the actual physical addresses using the identity of the node and knowledge of the system.

Attributes

ERR<n>ADDR is a 64-bit register.

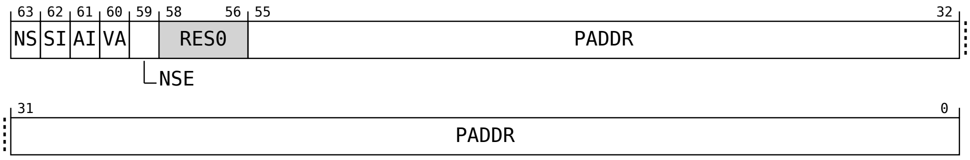
Configuration

rERR<q>FR describes the features implemented by the node that owns error record <n>. <q> is the index of the first error record owned by the same node as error record <n>. If the node owns a single record, then q = n.

This register is present only when error record is implemented and error record includes an address associated with an error. Otherwise, direct accesses to ERR<n>ADDR are RES0.

Field descriptions

The ERR<n>ADDR bit assignments are:



Bit [63]

When FEAT_RME is implemented

NS, bit [63]

Non-secure attribute. With ERR<n>ADDR.NSE, indicates the physical address space of the recorded location.

Value	Meaning
`0b0`	When ERR<n>ADDR.NSE == 0: ERR<n>ADDR.PADDR is a Secure address. When ERR<n>ADDR.NSE == 1: ERR<n>ADDR.PADDR is a Root address.
`0b1`	When ERR<n>ADDR.NSE == 0: ERR<n>ADDR.PADDR is a Non-secure address. When ERR<n>ADDR.NSE == 1: ERR<n>ADDR.PADDR is a Realm address.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When FEAT_RME is not implemented

NS, bit [63]

Non-secure attribute.

Value	Meaning
0b0	ERR<n>ADDR.PADDR is a Secure address.
0b1	ERR<n>ADDR.PADDR is a Non-secure address.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bit [62]

When FEAT_RME is implemented

SI, bit [62]

Secure Incorrect. Indicates whether ERR<n>ADDR.{NS, NSE} are valid.

Value	Meaning
0b0	ERR<n>ADDR.{NS, NSE} are correct. That is, they match the programmers' view of the physical address space for the recorded location.
0b1	ERR<n>ADDR.{NS, NSE} might not be correct, and might not match the programmers' view of the physical address space for the recorded location.

It is IMPLEMENTATION DEFINED whether this field is read-only or read/write.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When FEAT_RME is not implemented

SI, bit [62]

Secure Incorrect. Indicates whether ERR<n>ADDR.NS is valid.

Value	Meaning
0b0	ERR<n>ADDR.NS is correct. That is, it matches the programmers' view of the Non-secure attribute for the recorded location.
0b1	ERR<n>ADDR.NS might not be correct, and might not match the programmers' view of the Non-secure attribute for the recorded location.

It is IMPLEMENTATION DEFINED whether this field is read-only or read/write.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

AI, bit [61]

Address Incorrect. Indicates whether ERR<n>ADDR.PADDR is a valid physical address that is known to match the programmers' view of the physical address for the recorded location.

Value	Meaning
`0b0`	ERR<n>ADDR.PADDR is a valid physical address. That is, it matches the programmers' view of the physical address for the recorded location.
`0b1`	ERR<n>ADDR.PADDR might not be a valid physical address, and might not match the programmers' view of the physical address for the recorded location.

It is IMPLEMENTATION DEFINED whether this field is read-only or read/write.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

VA, bit [60]

Virtual Address. Indicates whether ERR<n>ADDR.PADDR field is a virtual address.

Value	Meaning
`0b0`	ERR<n>ADDR.PADDR is not a virtual address.
`0b1`	ERR<n>ADDR.PADDR is a virtual address.

No context information is provided for the virtual address. When ERR<n>ADDR.VA is 1, ERR<n>ADDR.{NS, SI, AI} read as {0, 1, 1}.

Support for this field is optional. If this field is not implemented and ERR<n>ADDR.PADDR field is a virtual address, then ERR<n>ADDR.{NS, SI, AI} read as {0, 1, 1}.

It is IMPLEMENTATION DEFINED whether this field is read-only or read/write.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

NSE, bit [59]

When FEAT_RME is implemented:

Physical Address Space. Together with ERR<n>ADDR.NS, indicates the address space for ERR<n>ADDR.PADDR.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [58:56]

Reserved, RES0.

PADDR, bits [55:0]

Physical Address. Address of the recorded location. If the physical address size implemented by this component is smaller than the size of this field, then high-order bits are unimplemented and either RES0 or have a fixed read-only IMPLEMENTATION DEFINED value. Low-order address bits might also be unimplemented and RES0, for example, if the physical address is always aligned to the size of a protection granule.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing the ERR<n>ADDR

ERR<n>ADDR can be accessed through the memory-mapped interface:

Component	Offset	Instance
RAS	0x018 + (64 * n)	ERR<n>ADDR

This interface is accessible as follows:

- When the Common Fault Injection Model Extension is implemented by the node that owns this error record, ERRPFGF.AV == 0 and ERRSTATUS.AV == 1 access to this register is **RO**.
- When the Common Fault Injection Model Extension is not implemented by the node that owns this error record and ERRSTATUS.AV == 1 access to this register is **RO**.
- Otherwise access to this register is **RW**.

A2.4.8 PMAUTHSTATUS, Performance Monitors Authentication Status register

The PMAUTHSTATUS characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for Performance Monitors.

Attributes

PMAUTHSTATUS is a 32-bit register.

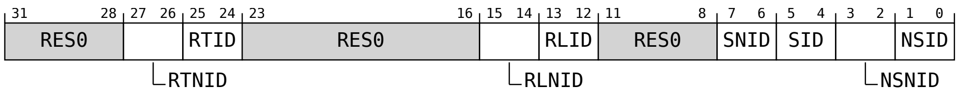
Configuration

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is OPTIONAL, and is required for CoreSight compliance. Arm recommends that this register is implemented.

Field descriptions

The PMAUTHSTATUS bit assignments are:



Bits [31:28]

Reserved, RES0.

RTNID, bits [27:26]

Root non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RTNID.

RTID, bits [25:24]

Root invasive debug.

Value	Meaning
`0b00`	Not implemented.

Bits [23:16]

Reserved, RES0.

RLNID, bits [15:14]

Realm non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RLNID.

RLID, bits [13:12]

Realm invasive debug.

Value	Meaning
`0b00`	Not implemented.

Bits [11:8]

Reserved, RES0.

SNID, bits [7:6]

Holds the same value as DBGAUTHSTATUS_EL1.SNID.

SID, bits [5:4]

Secure invasive debug. Possible values of this field are:

Value	Meaning
`0b00`	Not implemented.

All other values are reserved.

NSNID, bits [3:2]

Holds the same value as DBGAUTHSTATUS_EL1.NSNID.

NSID, bits [1:0]

Non-secure invasive debug. Possible values of this field are:

Value	Meaning
`0b00`	Not implemented.

All other values are reserved.

Accessing the PMAUTHSTATUS

PMAUTHSTATUS can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xFB8	PMAUTHSTATUS

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered() access to this register is **RO**.
- Otherwise access to this register returns an ERROR.

A2.4.9 PMCCFILTR_EL0, Performance Monitors Cycle Counter Filter Register

The PMCCFILTR_EL0 characteristics are:

Purpose

Determines the modes in which the Cycle Counter, PMCCNTR_EL0, increments.

Attributes

PMCCFILTR_EL0 is a 32-bit register.

Configuration

On a Warm or Cold reset, RW fields in this register reset to:

- Architecturally UNKNOWN values if the reset is to an Exception level that is using AArch64.
- 0 if the reset is to an Exception level that is using AArch32.

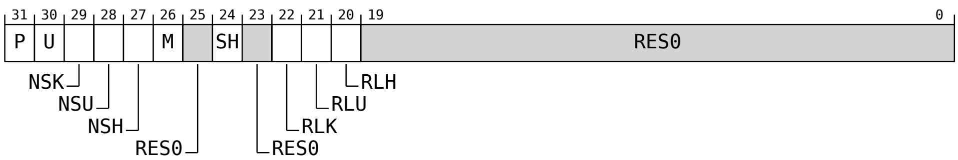
The register is not affected by an External debug reset.

External register PMCCFILTR_EL0 bits [31:0] are architecturally mapped to AArch64 system register PMCCFILTR_EL0[31:0].

External register PMCCFILTR_EL0 bits [31:0] are architecturally mapped to AArch32 system register PMCCFILTR[31:0].

Field descriptions

The PMCCFILTR_EL0 bit assignments are:



P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMCCFILTR_EL0.NSK bit.

If FEAT_RME is implemented, then counting in Realm EL1 is further controlled by the PMCCFILTR_EL0.RLK bit.

Value	Meaning
`0b0`	Count cycles in EL1.
`0b1`	Do not count cycles in EL1.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMCCFILTR_EL0.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMCCFILTR_EL0.RLU bit.

Value	Meaning
0b0	Count cycles in EL0.
0b1	Do not count cycles in EL0.

NSK, bit [29]

When EL3 is implemented:

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Non-secure EL1 are counted.

Otherwise, cycles in Non-secure EL1 are not counted.

Otherwise:

RES0

NSU, bit [28]

When EL3 is implemented:

Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Non-secure EL0 are counted.

Otherwise, cycles in Non-secure EL0 are not counted.

Otherwise:

RES0

NSH, bit [27]

When EL2 is implemented:

EL2 (Hypervisor) filtering bit. Controls counting in EL2.

If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMCCFILTR_EL0.SH bit.

If FEAT_RME is implemented, then counting in Realm EL2 is further controlled by the PMCCFILTR_EL0.RLH bit.

Value	Meaning
0b0	Do not count cycles in EL2.
0b1	Count cycles in EL2.

Otherwise:

RES0

M, bit [26]

When EL3 is implemented:

Secure EL3 filtering bit.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Secure EL3 are counted.

Otherwise, cycles in Secure EL3 are not counted.

Most applications can ignore this field and set its value to 0.

This field is not visible in the AArch32 PMCCFILTR System register.

Otherwise:

RES0

Bit [25]

Reserved, RES0.

SH, bit [24]

When FEAT_SEL2 is implemented and EL3 is implemented:

Secure EL2 filtering.

If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Secure EL2 are counted.

Otherwise, cycles in Secure EL2 are not counted.

If Secure EL2 is disabled, this field is RES0.

This field is not visible in the AArch32 PMCCFILTR System register.

Otherwise:

RES0

Bit [23]

Reserved, RES0.

RLK, bit [22]

When FEAT_RME is implemented:

Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Realm EL1 are counted.

Otherwise, cycles in Realm EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

RLU, bit [21]

When FEAT_RME is implemented:

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Realm EL0 are counted.

Otherwise, cycles in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

RLH, bit [20]

When FEAT_RME is implemented:

Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Realm EL2 are counted.

Otherwise, cycles in Realm EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [19:0]

Reserved, RES0.

Accessing the PMCCFILTR_EL0

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMCCFILTR_EL0 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0x47C	PMCCFILTR_EL0

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus() access to this register is **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus() access to this register is **RW**.
- Otherwise access to this register returns an ERROR.

A2.4.10 PMEVTYPER<n>_EL0, Performance Monitors Event Type Registers, n = 0 - 30

The PMEVTYPER<n>_EL0 characteristics are:

Purpose

Configures event counter n, where n is 0 to 30.

Attributes

PMEVTYPER<n>_EL0 is a 32-bit register.

Configuration

If event counter n is not implemented:

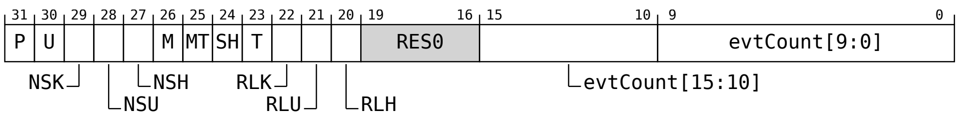
- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

External register PMEVTYPER<n>_EL0 bits [31:0] are architecturally mapped to AArch64 system register PMEVTYPER<n>_EL0[31:0].

External register PMEVTYPER<n>_EL0 bits [31:0] are architecturally mapped to AArch32 system register PMEVTYPER<n>[31:0].

Field descriptions

The PMEVTYPER<n>_EL0 bit assignments are:



P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit.

If FEAT_RME is implemented, then counting in Realm EL1 is further controlled by the PMEVTYPER<n>_EL0.RLK bit.

Value	Meaning
0b0	Count events in EL1.
0b1	Do not count events in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMEVTYPER<n>_EL0.RLU bit.

Value	Meaning
`0b0`	Count events in EL0.
`0b1`	Do not count events in EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSK, bit [29]

When EL3 is implemented:

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted.

Otherwise, events in Non-secure EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NSU, bit [28]

When EL3 is implemented:

Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted.

Otherwise, events in Non-secure EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NSH, bit [27]

When EL2 is implemented:

EL2 (Hypervisor) filtering bit. Controls counting in EL2.

If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit.

If FEAT_RME is implemented, then counting in Realm EL2 is further controlled by the PMEVTYPER<n>_EL0.RLH bit.

Value	Meaning
0b0	Do not count events in EL2.
0b1	Count events in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

M, bit [26]

When EL3 is implemented:

EL3 filtering bit.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted.

Otherwise, events in EL3 are not counted.

Most applications can ignore this field and set its value to 0b0.

This field is not visible in the AArch32 PMEVTYPER<n> System register.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

MT, bit [25]

When (FEAT_MTPMU is implemented and enabled) or an IMPLEMENTATION DEFINED multi-threaded PMU Extension is implemented:

Multithreading.

Value	Meaning
0b0	Count events only on controlling PE.
0b1	Count events from any PE with the same affinity at level 1 and above as this PE.

- When the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach, an implementation is described as multi-threaded. That is, the performance of PEs at the lowest affinity level is highly interdependent.
- Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

SH, bit [24]

When FEAT_SEL2 is implemented and EL3 is implemented:

Secure EL2 filtering.

If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted.

Otherwise, events in Secure EL2 are not counted.

This field is not visible in the AArch32 PMEVTYPER<n> System register.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

T, bit [23]

When FEAT_TME is implemented:

Transactional state filtering bit. Controls counting in Transactional state.

Value	Meaning
0b0	This bit has no effect on the filtering of events.
0b1	Do not count events in Transactional state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

RLK, bit [22]

When FEAT_RME is implemented:

Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted.

Otherwise, events in Realm EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

RLU, bit [21]

When FEAT_RME is implemented:

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted.

Otherwise, events in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

RLH, bit [20]

When FEAT_RME is implemented:

Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted.

Otherwise, events in Realm EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

Bits [19:16]

Reserved, RES0.

evtCount[15:10], bits [15:10]

When FEAT_PMUv3p1 is implemented:

Extension to evtCount[9:0]. For more information, see evtCount[9:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

evtCount[9:0], bits [9:0]

Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0.

Software must program this field with an event that is supported by the PE being programmed.

The ranges of event numbers allocated to each type of event are shown in 'Allocation of the PMU event number space'.

If FEAT_PMUv3p8 is implemented and PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.

Otherwise, if PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written:

- For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.
- If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.
- For other values, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is UNKNOWN.

UNPREDICTABLE means the event must not expose privileged information.

Arm recommends that for all values that represent reserved or unsupported events, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing the PMEVTYPER<n>_EL0

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMEVTYPER<n>_EL0 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0x400 + (4 * n)	PMEVTYPER<n>_EL0

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus() access to this register is **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus() access to this register is **RW**.
- Otherwise access to this register returns an ERROR.

A2.4.11 PMPCSR, Program Counter Sample Register

The PMPCSR characteristics are:

Purpose

Holds a sampled instruction address value.

Attributes

PMPCSR is a 64-bit register.

Configuration

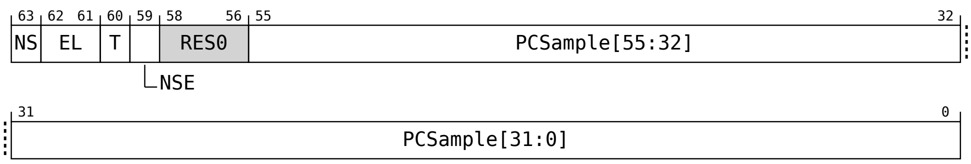
Before Armv8.2, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of EDDEVID.PCSample.

Support for 64-bit atomic reads is IMPLEMENTATION DEFINED. If 64-bit atomic reads are implemented, a 64-bit read of PMPCSR has the same side-effect as a 32-bit read of PMCSR[31:0] followed by a 32-bit read of PMPCSR[63:32], returning the combined value. For example, if the PE is in Debug state then a 64-bit atomic read returns bits[31:0] == 0xFFFFFFFF and bits[63:32] UNKNOWN.

This register is present only when FEAT_PCSRv8p2 is implemented. Otherwise, direct accesses to PMPCSR are RES0.

Field descriptions

The PMPCSR bit assignments are:



Bit [63]

When FEAT_RME is implemented

NS, bit [63]

Together with the NSE field, indicates the Security state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

NSE	NS	Meaning
0b0	0b0	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Otherwise

NS, bit [63]

Non-secure state sample. Indicates the Security state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

If EL3 is not implemented, this bit indicates the Effective value of SCR.NS.

Value	Meaning
0b0	Sample is from Secure state.
0b1	Sample is from Non-secure state.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

EL, bits [62:61]

Exception level status sample. Indicates the Exception level that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

Value	Meaning
0b00	Sample is from EL0.
0b01	Sample is from EL1.
0b10	Sample is from EL2.
0b11	Sample is from EL3.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

T, bit [60]

When FEAT_TME is implemented:

Transactional state of the sample. Indicates the Transactional state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

Value	Meaning
0b0	Sample is from Non-transactional state.
0b1	Sample is from Transactional state.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RES0

NSE, bit [59]

When FEAT_RME is implemented:

Together with the NS field, indicates the Security state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

For a description of the values derived by evaluating NS and NSE together, see PMPCSR.NS.

Otherwise:

RES0

Bits [58:56]

Reserved, RES0.

PCSample[55:32], bits [55:32]

Bits[55:32] of the sampled instruction address value. The translation regime that PMPCSR samples can be determined from PMPCSR.{NS,EL}.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

PCSample[31:0], bits [31:0]

Bits[31:0] of the sampled instruction address value.

PMPCSR[31:0] reads as 0xFFFFFFFF when any of the following are true:

- The PE is in Debug state.
- PC Sample-based profiling is prohibited.

If a branch instruction has retired since the PE left reset state, then the first read of PMPCSR[31:0] is permitted but not required to return 0xFFFFFFFF.

PMPCSR[31:0] reads as an UNKNOWN value when any of the following are true:

- The PE is in reset state.
- No branch instruction has retired since the PE left reset state, Debug state, or a state where PC Sample-based Profiling is prohibited.
- No branch instruction has retired since the last read of PMPCSR[31:0].

For the cases where a read of PMPCSR[31:0] returns 0xFFFFFFFF or an UNKNOWN value, the read has the side-effect of setting PMPCSR[63:32], PMCID1SR, PMCID2SR, and PMVIDSR to UNKNOWN values.

Otherwise, a read of PMPCSR[31:0] returns bits [31:0] of the sampled instruction address value and has the side-effect of indirectly writing to PMPCSR[63:32], PMCID1SR, PMCID2SR, and PMVIDSR. The translation regime that PMPCSR samples can be determined from PMPCSR.{NS,EL}.

For a read of PMPCSR[31:0] from the memory-mapped interface, if PMLSR.SLK == 1, meaning the OPTIONAL Software Lock is locked, then the side-effect of the access does not occur and PMPCSR[63:32], PMCID1SR, PMCID2SR, and PMVIDSR are unchanged.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing the PMPCSR

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

PMPCSR can be accessed through the external debug interface:

Component	Offset	Instance	Range
PMU	0x200	PMPCSR	31:0

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus() access to this register is **RO**.
- Otherwise access to this register returns an ERROR.

PMPCSR can be accessed through the external debug interface:

Component	Offset	Instance	Range
PMU	0x204	PMPCSR	63:32

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus() access to this register is **RO**.
- Otherwise access to this register returns an ERROR.

PMPCSR can be accessed through the external debug interface:

Component	Offset	Instance	Range
PMU	0x220	PMPCSR	31:0

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus() access to this register is **RO**.
- Otherwise access to this register returns an ERROR.

PMPCSR can be accessed through the external debug interface:

Component	Offset	Instance	Range
PMU	0x224	PMPCSR	63:32

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus() access to this register is **RO**.
- Otherwise access to this register returns an ERROR.

A2.4.12 TRCAUTHSTATUS, Authentication Status Register

The TRCAUTHSTATUS characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

For additional information, see the CoreSight Architecture Specification.

Attributes

TRCAUTHSTATUS is a 32-bit register.

Configuration

External register TRCAUTHSTATUS bits [31:0] are architecturally mapped to AArch64 system register TRCAUTHSTATUS[31:0].

This register is present only when FEAT_ETE is implemented. Otherwise, direct accesses to TRCAUTHSTATUS are RES0.

Field descriptions

The TRCAUTHSTATUS bit assignments are:

31:28	27:26	25:24	23:16	15:14	13:12	11:10	9:8	7:6	5:4	3:2	1:0
RES0	RTNID	RTID	RES0	RLNID	RLID	HNID	HID	SNID	SID	NSNID	NSID

Bits [31:28]

Reserved, RES0.

RTNID, bits [27:26]

Root non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RTNID.

RTID, bits [25:24]

Root invasive debug.

Value	Meaning
`0b00`	Not implemented.

Bits [23:16]

Reserved, RES0.

RLNID, bits [15:14]

Realm non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RLNID.

RLID, bits [13:12]

Realm invasive debug.

Value	Meaning
`0b00`	Not implemented.

HNID, bits [11:10]

Hyp Non-invasive Debug. Indicates whether a separate enable control for EL2 non-invasive debug features is implemented and enabled.

Value	Meaning
`0b00`	Separate Hyp non-invasive debug enable not implemented, or EL2 non-invasive debug features not implemented.
`0b10`	Implemented and disabled.
`0b11`	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

HID, bits [9:8]

Hyp Invasive Debug. Indicates whether a separate enable control for EL2 invasive debug features is implemented and enabled.

Value	Meaning
`0b00`	Separate Hyp invasive debug enable not implemented, or EL2 invasive debug features not implemented.
`0b10`	Implemented and disabled.
`0b11`	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

SNID, bits [7:6]

Secure Non-invasive Debug. Indicates whether Secure non-invasive debug features are implemented and enabled.

Value	Meaning
`0b00`	Secure non-invasive debug features not implemented.
`0b10`	Implemented and disabled.

Value	Meaning
`0b11`	Implemented and enabled.

All other values are reserved.

When EL3 is implemented, this field takes the value 0b10 or 0b11 depending whether Secure non-invasive debug is enabled.

When EL3 is not implemented and the PE is Non-secure, this field reads as 0b00.

When EL3 is not implemented and the PE is Secure, this field takes the value 0b10 or 0b11 depending whether Secure non-invasive debug is enabled.

SID, bits [5:4]

Secure Invasive Debug. Indicates whether Secure invasive debug features are implemented and enabled.

Value	Meaning
`0b00`	Secure invasive debug features not implemented.
`0b10`	Implemented and disabled.
`0b11`	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

NSNID, bits [3:2]

Non-secure Non-invasive Debug. Indicates whether Non-secure non-invasive debug features are implemented and enabled.

Value	Meaning
`0b00`	Non-secure non-invasive debug features not implemented.
`0b10`	Implemented and disabled.
`0b11`	Implemented and enabled.

All other values are reserved.

When EL3 is implemented, this field reads as 0b11.

When EL3 is not implemented and the PE is Non-secure, this field reads as 0b11.

When EL3 is not implemented and the PE is Secure, this field reads as 0b00.

NSID, bits [1:0]

Non-secure Invasive Debug. Indicates whether Non-secure invasive debug features are implemented and enabled.

Value	Meaning
0b00	Non-secure invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

Accessing the TRCAUTHSTATUS

For implementations that support multiple access mechanisms, different access mechanisms can return different values for reads of TRCAUTHSTATUS if the authentication signals have changed and that change has not yet been synchronized by a Context synchronization event. This scenario can happen if, for example, the external debugger view is implemented separately from the system instruction view to allow for separate power domains, and so observes changes on the signals differently.

External debugger accesses to this register are unaffected by the OS Lock.

TRCAUTHSTATUS can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFB8	TRCAUTHSTATUS

This interface is accessible as follows:

- When !IsTraceCorePowered() access to this register returns an ERROR.
- Otherwise access to this register is **RO**.

Chapter A3
List of instructions

This section provides the full information for instructions added or modified by RME.

A3.1 AArch64 System instructions

This section provides the full information for AArch64 System instructions added or modified by RME.

A3.1.1 CFP RCTX, Control Flow Prediction Restriction by Context

The CFP RCTX characteristics are:

Purpose

Control Flow Prediction Restriction by Context applies to all Control Flow Prediction Resources that predict execution based on information gathered within the target execution context or contexts.

Control flow predictions determined by the actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control speculative execution occurring after the instruction is complete and synchronized.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

This instruction does not require the invalidation of prediction structures so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Attributes

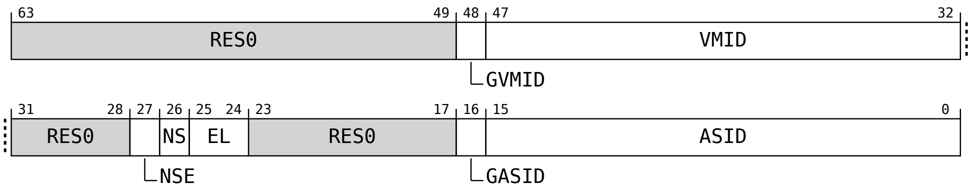
CFP RCTX is a 64-bit System instruction.

Configuration

This instruction is present only when FEAT_SPECRES is implemented. Otherwise, direct accesses to CFP RCTX are UNDEFINED.

Field descriptions

The CFP RCTX bit assignments are:



Bits [63:49]

Reserved, RES0.

GVMID, bit [48]

Execution of this instruction applies to all VMIDs or a specified VMID.

Value	Meaning
`0b0`	Applies to specified VMID for an EL0 or EL1 target execution context.
`0b1`	Applies to all VMIDs for an EL0 or EL1 target execution context.

For target execution contexts other than EL0 or EL1, this field is RES0.

If the instruction is executed at EL0 or EL1, this field has an Effective value of 0.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

VMID, bits [47:32]

Only applies when bit[48] is 0 and the target execution context is either:

- EL1.
- EL0 when (HCR_EL2.E2H==0 or HCR_EL2.TGE==0).

Otherwise this field is RES0.

When the instruction is executed at EL1, this field is treated as the current VMID.

When the instruction is executed at EL0 and (HCR_EL2.E2H==0 or HCR_EL2.TGE==0), this field is treated as the current VMID.

When the instruction is executed at EL0 and (HCR_EL2.E2H==1 and HCR_EL2.TGE==1), this field is ignored.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

If the implementation supports 16 bits of VMID, then the upper 8 bits of the VMID must be written to 0 by software when the context being affected only uses 8 bits.

Bits [31:28]

Reserved, RES0.

NSE, bit [27]

When FEAT_RME is implemented:

Together with the NS field, selects the Security state.

For a description of the values derived by evaluating NS and NSE together, see CFP_RCTX.NS.

Otherwise:

RES0

Bit [26]

When FEAT_RME is implemented

NS, bit [26]

Together with the NSE field, selects the Security state. Defined values are:

NSE	NS	Meaning
0b0	0b0	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Some Effective values are determined by the current Security state:

- When executed in Secure state, the Effective value of NSE is 0.

- When executed in Non-secure state, the Effective value of {NSE, NS} is {0, 1}.
- When executed in Realm state, the Effective value of {NSE, NS} is {1, 1}.

An instruction with an EL field that has a value other than 0b11 (EL3) is treated as a NOP when executed at EL3 with CFP_RCTX.{NSE, NS} == {1, 0}.

Otherwise

NS, bit [26]

Security State. Defined values are:

Value	Meaning
0b0	Secure state.
0b1	Non-secure state.

When executed in Non-secure state, the Effective value of NS is 1.

EL, bits [25:24]

Exception Level. Indicates the Exception level of the target execution context.

Value	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

If the instruction is executed at an Exception level lower than the specified level, this instruction is treated as a NOP.

Bits [23:17]

Reserved, RES0.

GASID, bit [16]

Execution of this instruction applies to all ASIDs or a specified ASID.

Value	Meaning
0b0	Applies to specified ASID for an EL0 target execution context.
0b1	Applies to all ASID for an EL0 target execution context.

For target execution contexts other than EL0, this field is RES0.

If the instruction is executed at EL0, this field has an Effective value of 0.

ASID, bits [15:0]

Only applies for an EL0 target execution context and when bit[16] is 0.

Otherwise, this field is RES0.

When the instruction is executed at EL0, this field is treated as the current ASID.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being affected only uses 8 bits.

Accessing the CFP RCTX

Accesses to this instruction use the following encodings in the instruction encoding space:

CFP RCTX, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0011	0b100

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.EnRCTX == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
     ↪HFGITR_EL2.CFPRCTX == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.EnRCTX == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.RestrictPrediction(X[t], RestrictType_ControlFlow);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.CFPRCTX == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.RestrictPrediction(X[t], RestrictType_ControlFlow);
elsif PSTATE.EL == EL2 then
    AArch64.RestrictPrediction(X[t], RestrictType_ControlFlow);
elsif PSTATE.EL == EL3 then
    AArch64.RestrictPrediction(X[t], RestrictType_ControlFlow);
```

A3.1.2 CPP RCTX, Cache Prefetch Prediction Restriction by Context

The CPP RCTX characteristics are:

Purpose

Cache Prefetch Prediction Restriction by Context applies to all Cache Allocation Resources that predict cache allocations based on information gathered within the target execution context or contexts.

Cache prefetch predictions determined by the actions of code in the target execution context or contexts appearing in program order before the instruction cannot influence speculative execution occurring after the instruction is complete and synchronized.

This instruction applies to all:

- Instruction caches.
- Data caches.
- TLB prefetching hardware used by the executing PE that applies to the supplied context or contexts.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

This instruction does not require the invalidation of Cache Allocation Resources so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Attributes

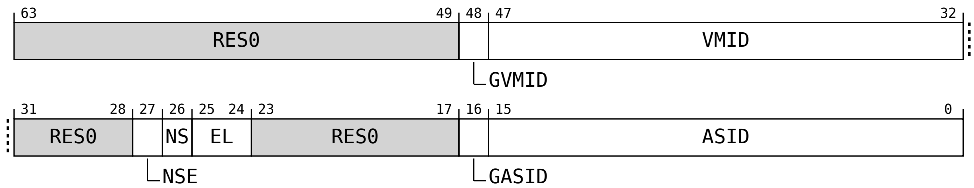
CPP RCTX is a 64-bit System instruction.

Configuration

This instruction is present only when FEAT_SPECRES is implemented. Otherwise, direct accesses to CPP RCTX are UNDEFINED.

Field descriptions

The CPP RCTX bit assignments are:



Bits [63:49]

Reserved, RES0.

GVMID, bit [48]

Execution of this instruction applies to all VMIDs or a specified VMID.

Value	Meaning
`0b0`	Applies to specified VMID for an EL0 or EL1 target execution context.
`0b1`	Applies to all VMIDs for an EL0 or EL1 target execution context.

For target execution contexts other than EL0 and EL1, this field is RES0.

If the instruction is executed at EL0 or EL1, this field has an Effective value of 0.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

VMID, bits [47:32]

Only applies when bit[48] is 0 and the target execution context is either:

- EL1.
- EL0 when (HCR_EL2.E2H==0 or HCR_EL2.TGE==0).

Otherwise this field is RES0.

When the instruction is executed at EL1, this field is treated as the current VMID.

When the instruction is executed at EL0 and (HCR_EL2.E2H==0 or HCR_EL2.TGE==0), this field is treated as the current VMID.

When the instruction is executed at EL0 and (HCR_EL2.E2H==1 and HCR_EL2.TGE==1), this field is ignored.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

If the implementation supports 16 bits of VMID, then the upper 8 bits of the VMID must be written to 0 by software when the context being affected only uses 8 bits.

Bits [31:28]

Reserved, RES0.

NSE, bit [27]

When FEAT_RME is implemented:

Together with the NS field, selects the Security state.

For a description of the values derived by evaluating NS and NSE together, see CPP_RCTX.NS.

Otherwise:

RES0

Bit [26]

When FEAT_RME is implemented

NS, bit [26]

Together with the NSE field, selects the Security state. Defined values are:

NSE	NS	Meaning
0b0	0b0	Secure.

NSE	NS	Meaning
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Some Effective values are determined by the current Security state:

- When executed in Secure state, the Effective value of NSE is 0.
- When executed in Non-secure state, the Effective value of {NSE, NS} is {0, 1}.
- When executed in Realm state, the Effective value of {NSE, NS} is {1, 1}.

An instruction with an EL field that has a value other than 0b11 (EL3) is treated as a NOP when executed at EL3 with CPP_RCTX.{NSE, NS} == {1, 0}.

Otherwise

NS, bit [26]

Security State. Defined values are:

Value	**Meaning**
`0b0`	Secure state.
`0b1`	Non-secure state.

When executed in Non-secure state, the Effective value of NS is 1.

EL, bits [25:24]

Exception Level. Indicates the Exception level of the target execution context.

Value	**Meaning**
`0b00`	EL0.
`0b01`	EL1.
`0b10`	EL2.
`0b11`	EL3.

If the instruction is executed at an Exception level lower than the specified level, this instruction is treated as a NOP.

Bits [23:17]

Reserved, RES0.

GASID, bit [16]

Execution of this instruction applies to all ASIDs or a specified ASID.

Value	Meaning
0b0	Applies to specified ASID for an EL0 target execution context.
0b1	Applies to all ASID for an EL0 target execution context.

For target execution contexts other than EL0, this field is RES0.

If the instruction is executed at EL0, this field has an Effective value of 0.

ASID, bits [15:0]

Only applies for an EL0 target execution context and when bit[16] is 0.

Otherwise, this field is RES0.

When the instruction is executed at EL0, this field is treated as the current ASID.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being affected only uses 8 bits.

Accessing the CPP RCTX

Accesses to this instruction use the following encodings in the instruction encoding space:

CPP RCTX, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0011	0b111

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.EnRCTX == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.CPPRCTX == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.EnRCTX == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.RestrictPrediction(X[t], RestrictType_CachePrefetch);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.CPPRCTX == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.RestrictPrediction(X[t], RestrictType_CachePrefetch);
elsif PSTATE.EL == EL2 then
    AArch64.RestrictPrediction(X[t], RestrictType_CachePrefetch);
elsif PSTATE.EL == EL3 then
    AArch64.RestrictPrediction(X[t], RestrictType_CachePrefetch);
```

A3.1.3 DC CIGDPAPA, Clean and Invalidate of Data and Allocation Tags by PA to PoPA

The DC CIGDPAPA characteristics are:

Purpose

Clean and Invalidate data and Allocation Tags in data cache by physical address to the Point of Physical Aliasing.

Attributes

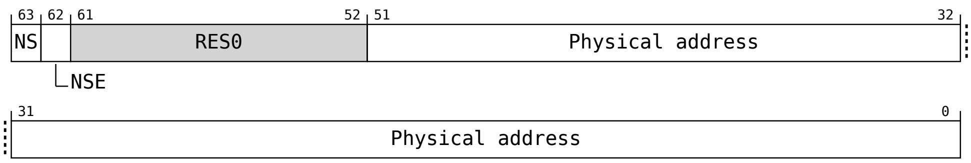
DC CIGDPAPA is a 64-bit System instruction.

Configuration

This instruction is present only when FEAT_RME is implemented and FEAT_MTE2 is implemented. Otherwise, direct accesses to DC CIGDPAPA are UNDEFINED.

Field descriptions

The DC CIGDPAPA bit assignments are:



NS, bit [63]

Together with the NSE field, this field specifies the target physical address space.

NSE	NS	Meaning
0b0	0b0	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

NSE, bit [62]

Together with the NS field, this field specifies the target physical address space.

For a description of the values derived by evaluating NS and NSE together, see DC CIGDPAPA.NS.

Bits [61:52]

Reserved, RES0.

Bits [51:0]

Physical address to use. No alignment restrictions apply to this PA.

Accessing the DC CIGDPAPA

- This instruction is not subject to any translation, permission checks, or granule protection checks.

- This instruction affects all caches in the Outer Shareable shareability domain.
- This instruction has the same ordering, observability, and completion behavior as VA-based cache maintenance instructions issued to the Outer Shareable shareability domain.

Accesses to this instruction use the following encodings in the instruction encoding space:

DC CIGDPAPA, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b110	0b0111	0b1110	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    DC_CIGDPAPA(X[t]);
```

A3.1.4 DC CIPAPA, Data or unified Cache line Clean and Invalidate by PA to PoPA

The DC CIPAPA characteristics are:

Purpose

Clean and Invalidate data cache by physical address to the Point of Physical Aliasing.

Attributes

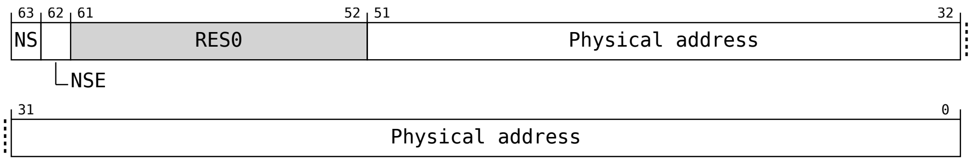
DC CIPAPA is a 64-bit System instruction.

Configuration

This instruction is present only when FEAT_RME is implemented. Otherwise, direct accesses to DC CIPAPA are UNDEFINED.

Field descriptions

The DC CIPAPA bit assignments are:



NS, bit [63]

Together with the NSE field, this field specifies the target physical address space.

NSE	NS	Meaning
0b0	0b0	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

NSE, bit [62]

Together with the NS field, this field specifies the target physical address space.

For a description of the values derived by evaluating NS and NSE together, see DC CIPAPA.NS.

Bits [61:52]

Reserved, RES0.

Bits [51:0]

Physical address to use. No alignment restrictions apply to this PA.

Accessing the DC CIPAPA

- This instruction is not subject to any translation, permission checks, or granule protection checks.
- This instruction affects all caches in the Outer Shareable shareability domain.

- This instruction has the same ordering, observability, and completion behavior as VA-based cache maintenance instructions issued to the Outer Shareable shareability domain.

Accesses to this instruction use the following encodings in the instruction encoding space:

DC CIPAPA, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b110	0b0111	0b1110	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    DC_CIPAPA(X[t]);
```


A3.1.5 DVP RCTX, Data Value Prediction Restriction by Context

The DVP RCTX characteristics are:

Purpose

Data Value Prediction Restriction by Context applies to all Data Value Prediction Resources that predict execution based on information gathered within the target execution context or contexts.

Data value predictions determined by the actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control speculative execution occurring after the instruction is complete and synchronized.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

This instruction does not require the invalidation of prediction structures so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Attributes

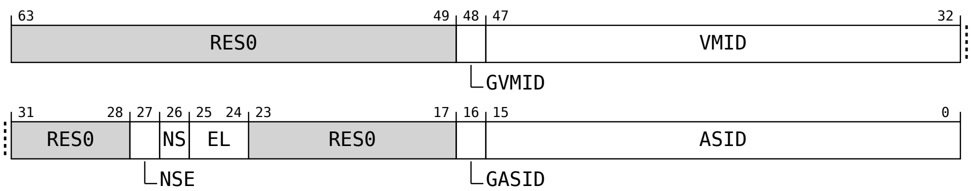
DVP RCTX is a 64-bit System instruction.

Configuration

This instruction is present only when FEAT_SPECRES is implemented. Otherwise, direct accesses to DVP RCTX are UNDEFINED.

Field descriptions

The DVP RCTX bit assignments are:



Bits [63:49]

Reserved, RES0.

GVMID, bit [48]

Execution of this instruction applies to all VMIDs or a specified VMID.

Value	Meaning
`0b0`	Applies to specified VMID for an EL0 or EL1 target execution context.
`0b1`	Applies to all VMIDs for an EL0 or EL1 target execution context.

For target execution contexts other than EL0 or EL1, this field is RES0.

If the instruction is executed at EL0 or EL1, then this field has an Effective value of 0.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

VMID, bits [47:32]

Only applies when bit[48] is 0 and the target execution context is either:

- EL1.
- EL0 when (HCR_EL2.E2H==0 or HCR_EL2.TGE==0).

Otherwise this field is RES0.

When the instruction is executed at EL1, this field is treated as the current VMID.

When the instruction is executed at EL0 and (HCR_EL2.E2H==0 or HCR_EL2.TGE==0), this field is treated as the current VMID.

When the instruction is executed at EL0 and (HCR_EL2.E2H==1 and HCR_EL2.TGE==1), this field is ignored.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

If the implementation supports 16 bits of VMID, then the upper 8 bits of the VMID must be written to 0 by software when the context being affected only uses 8 bits.

Bits [31:28]

Reserved, RES0.

NSE, bit [27]

When FEAT_RME is implemented:

Together with the NS field, selects the Security state.

For a description of the values derived by evaluating NS and NSE together, see DVP_RCTX.NS.

Otherwise:

RES0

Bit [26]

When FEAT_RME is implemented

NS, bit [26]

Together with the NSE field, selects the Security state. Defined values are:

NSE	NS	Meaning
0b0	0b0	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Some Effective values are determined by the current Security state:

- When executed in Secure state, the Effective value of NSE is 0.

- When executed in Non-secure state, the Effective value of {NSE, NS} is {0, 1}.
- When executed in Realm state, the Effective value of {NSE, NS} is {1, 1}.

An instruction with an EL field that has a value other than 0b11 (EL3) is treated as a NOP when executed at EL3 with DVP_RCTX.{NSE, NS} == {1, 0}.

Otherwise

NS, bit [26]

Security State. Defined values are:

Value	Meaning
0b0	Secure state.
0b1	Non-secure state.

When executed in Non-secure state, the Effective value of NS is 1.

EL, bits [25:24]

Exception Level. Indicates the Exception level of the target execution context.

Value	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

If the instruction is executed at an Exception level lower than the specified level, this instruction is treated as a NOP.

Bits [23:17]

Reserved, RES0.

GASID, bit [16]

Execution of this instruction applies to all ASIDs or a specified ASID.

Value	Meaning
0b0	Applies to specified ASID for an EL0 target execution context.
0b1	Applies to all ASID for an EL0 target execution context.

For target execution contexts other than EL0, this field is RES0.

If the instruction is executed at EL0, this field has an Effective value of 0.

ASID, bits [15:0]

Only applies for an EL0 target execution context and when bit[16] is 0.

Otherwise this field is RES0.

When the instruction is executed at EL0, this field is treated as the current ASID.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being affected only uses 8 bits.

Accessing the DVP RCTX

Accesses to this instruction use the following encodings in the instruction encoding space:

DVP RCTX, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0011	0b101

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.EnRCTX == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
        ↪HFGITR_EL2.DVPRCTX == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.EnRCTX == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.RestrictPrediction(X[t], RestrictType_DataValue);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DVPRCTX == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.RestrictPrediction(X[t], RestrictType_DataValue);
elsif PSTATE.EL == EL2 then
    AArch64.RestrictPrediction(X[t], RestrictType_DataValue);
elsif PSTATE.EL == EL3 then
    AArch64.RestrictPrediction(X[t], RestrictType_DataValue);
```

A3.1.6 TLBI PAALL, TLB Invalidate GPT Information by PA, All Entries, Local

The TLBI PAALL characteristics are:

Purpose

Invalidates cached copies of GPT entries from TLBs. Details:

- The invalidation applies to TLB entries containing GPT information that relates to a physical address.
- The invalidation applies to all TLB entries containing GPT information.
- The invalidation affects only the TLBs for the PE executing the operation.

The full set of TLB maintenance instructions that invalidate cached GPT entries is: TLBI PAALL, TLBI PAALLOS, TLBI RPALOS, and TLBI RPAOS.

These instructions have the same ordering, observability, and completion behavior as all other TLBI instructions.

Configuration

This instruction is present only when FEAT_RME is implemented. Otherwise, direct accesses to TLBI PAALL are UNDEFINED.

Accessing the TLBI PAALL

Accesses to this instruction use the following encodings in the instruction encoding space:

TLBI PAALL{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0111	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_PAALL(Shareability_NSH);
```

A3.1.7 TLBI PAALLOS, TLB Invalidate GPT Information by PA, All Entries, Outer Shareable

The TLBI PAALLOS characteristics are:

Purpose

Invalidates cached copies of GPT entries from TLBs. Details:

- The invalidation applies to TLB entries containing GPT information that relates to a physical address.
- The invalidation applies to all TLB entries containing GPT information.
- The invalidation affects all TLBs in the Outer Shareable domain.

The full set of TLB maintenance instructions that invalidate cached GPT entries is: TLBI PAALL, TLBI PAALLOS, TLBI RPALOS, and TLBI RPAOS.

These instructions have the same ordering, observability, and completion behavior as all other TLBI instructions.

Configuration

This instruction is present only when FEAT_RME is implemented. Otherwise, direct accesses to TLBI PAALLOS are UNDEFINED.

Accessing the TLBI PAALLOS

Accesses to this instruction use the following encodings in the instruction encoding space:

TLBI PAALLOS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0001	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_PAALL(Shareability_OSH);
```

A3.1.8 TLBI RPALOS, TLB Range Invalidate GPT Information by PA, Last level, Outer Shareable

The TLBI RPALOS characteristics are:

Purpose

Invalidates cached copies of GPT entries from TLBs. Details:

- The invalidation applies to TLB entries containing GPT information that relates to a physical address.
- The invalidation affects all TLBs in the Outer Shareable domain.
- Invalidates TLB entries containing GPT information from the final level of the GPT walk that relates to the supplied physical address.
- Invalidations are range-based, invalidating TLB entries starting from the address in BaseADDR, within the range as specified by SIZE.

The full set of TLB maintenance instructions that invalidate cached GPT entries is: TLBI PAALL, TLBI PAALLOS, TLBI RPALOS, and TLBI RPAOS.

These instructions have the same ordering, observability, and completion behavior as all other TLBI instructions.

Attributes

TLBI RPALOS is a 64-bit System instruction.

Configuration

This instruction is present only when FEAT_RME is implemented. Otherwise, direct accesses to TLBI RPALOS are UNDEFINED.

Field descriptions

The TLBI RPALOS bit assignments are:

63–48	47–44	43–40	39–32
RES0	SIZE	RES0	Address

31–0
Address

Bits [63:48]

Reserved, RES0.

SIZE, bits [47:44]

Size of the range for invalidation.

If SIZE is a reserved value, no TLB entries are required to be invalidated.

Value	Meaning
0b0000	4KB.
0b0001	16KB.
0b0010	64KB.

Value	Meaning
`0b0011`	2MB.
`0b0100`	32MB.
`0b0101`	512MB.
`0b0110`	1GB.
`0b0111`	16GB.
`0b1000`	64GB.
`0b1001`	512GB.

All other values are reserved.

If SIZE gives a range smaller than the configured physical granule size in GPCCR_EL3.PGS, then the effective value of SIZE is taken to be the size configured by GPCCR_EL3.PGS.

If GPCCR_EL3.PGS is configured to a reserved value, no TLB entries are required to be invalidated.

Bits [43:40]

Reserved, RES0.

Address, bits [39:0]

The starting address for the range of the maintenance instruction.

This field is decoded with reference to the value of GPCCR_EL3.PGS to give BaseADDR as follows:

GPCCR_EL3.PGS	BaseADDR
0b00 (4KB)	BaseADDR[51:12] = Xt[39:0]
0b10 (16KB)	BaseADDR[51:14] = Xt[39:2]
0b01 (64KB)	BaseADDR[51:16] = Xt[39:4]

Other bits of BaseADDR are treated as zero, to give the effective value of BaseADDR.

If the effective value of BaseADDR is not aligned to the size of the effective value of SIZE, no TLB entries are required to be invalidated.

If GPCCR_EL3.PGS is configured to a reserved value, no TLB entries are required to be invalidated.

Accessing the TLBI RPALOS

Accesses to this instruction use the following encodings in the instruction encoding space:

TLBI RPALOS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0100	0b111

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RPA(TLBILevel_Last, X[t], Shareability_OSH);
```

A3.1.9 TLBI RPAOS, TLB Range Invalidate GPT Information by PA, Outer Shareable

The TLBI RPAOS characteristics are:

Purpose

Invalidates cached copies of GPT entries from TLBs. Details:

- The invalidation applies to TLB entries containing GPT information that relates to a physical address.
- The invalidation affects all TLBs in the Outer Shareable domain.
- Invalidates TLB entries containing GPT information from all levels of the GPT walk that relates to the supplied physical address.
- Invalidations are range-based, invalidating TLB entries starting from the address in BaseADDR, within the range as specified by SIZE.

The full set of TLB maintenance instructions that invalidate cached GPT entries is: TLBI PAALL, TLBI PAALLOS, TLBI RPALOS, and TLBI RPAOS.

These instructions have the same ordering, observability, and completion behavior as all other TLBI instructions.

Attributes

TLBI RPAOS is a 64-bit System instruction.

Configuration

This instruction is present only when FEAT_RME is implemented. Otherwise, direct accesses to TLBI RPAOS are UNDEFINED.

Field descriptions

The TLBI RPAOS bit assignments are:

63 - 48	47 - 44	43 - 40	39 - 32
RES0	SIZE	RES0	Address

31 - 0
Address

Bits [63:48]

Reserved, RES0.

SIZE, bits [47:44]

Size of the range for invalidation.

If SIZE is a reserved value, no TLB entries are required to be invalidated.

Value	Meaning
`0b0000`	4KB.
`0b0001`	16KB.
`0b0010`	64KB.
`0b0011`	2MB.

Value	Meaning
0b0100	32MB.
0b0101	512MB.
0b0110	1GB.
0b0111	16GB.
0b1000	64GB.
0b1001	512GB.

All other values are reserved.

If SIZE gives a range smaller than the configured physical granule size in GPCCR_EL3.PGS, then the effective value of SIZE is taken to be the size configured by GPCCR_EL3.PGS.

If GPCCR_EL3.PGS is configured to a reserved value, no TLB entries are required to be invalidated.

Bits [43:40]

Reserved, RES0.

Address, bits [39:0]

The starting address for the range of the maintenance instruction.

This field is decoded with reference to the value of GPCCR_EL3.PGS to give BaseADDR as follows:

GPCCR_EL3.PGS	BaseADDR
0b00 (4KB)	BaseADDR[51:12] = Xt[39:0]
0b10 (16KB)	BaseADDR[51:14] = Xt[39:2]
0b01 (64KB)	BaseADDR[51:16] = Xt[39:4]

Other bits of BaseADDR are treated as zero, to give the effective value of BaseADDR.

If the effective value of BaseADDR is not aligned to the size of the effective value of SIZE, no TLB entries are required to be invalidated.

If GPCCR_EL3.PGS is configured to a reserved value, no TLB entries are required to be invalidated.

Accessing the TLBI RPAOS

Accesses to this instruction use the following encodings in the instruction encoding space:

TLBI RPAOS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0100	0b011

```
if PSTATE.EL == EL0 then
    UNDEFINED;
```

```
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RPA(TLBILevel_Any, X[t], Shareability_OSH);
```

Glossary

BRBE

Branch Record Buffer Extension.

Feature introduced in Armv9.2-A, see [1].

GPC

Granule Protection Check

GPC exception

Granule Protection Check exception

Class of synchronous exceptions, triggered by a GPC fault.

GPC fault

Granule Protection Check fault

Any fault returned by a granule protection check, which can be one of:

- Granule protection fault.
- GPT walk fault.
- GPT address size fault.
- Synchronous External abort on GPT fetch.

GPF

Granule Protection Fault

A type of GPC fault. An access fails a granule protection check with a GPF when the GPT lookup succeeds and the GPT information does not permit the access.

GPT

Granule Protection Table

The in-memory structure that describes the association of physical granules with physical address spaces.

IRI

Interrupt Routing Infrastructure. The part of the GIC comprising the Distributor, Redistributors and ITSs.

PAS

Physical Address Space

PoPA

Point of Physical Aliasing

See 4.4.1 *Point of Physical Aliasing* for more information.