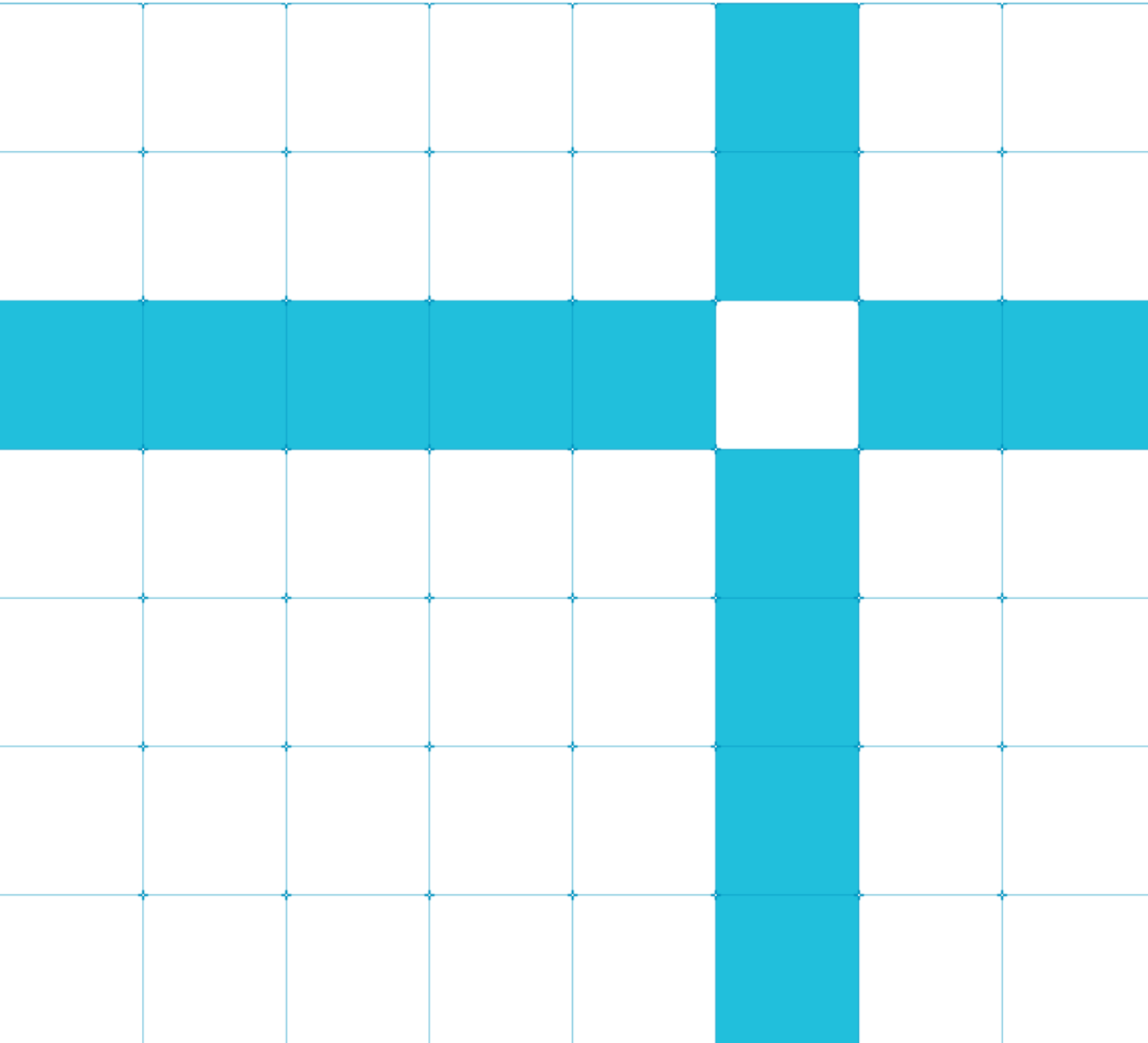
Arm® Cortex®-M35P r1p1 Lite Security Target

Version 1.1



Cortex®-M35P

Lite Security Target

Release Information

Document History

Version	Date	Confidentiality	Change
1.0	15 November 2019	Non-Confidential	First draft for r1p1
1.1	16 January 2020	Non-Confidential	First release for r1p1

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

http://www.arm.com

Contents

1 About this document

1.1. Scope

This document identifies the security properties of the TOE and defines the scope of the evaluation.

1.2. References

This document refers to the following publications:

Document	Title
[PP84]	*Security IC Platform Protection Profile with Augmentation Packages, BSI-CC-PP-0084-2014, version 1.0, date 3.01.2014*
[CC]	*Common Criteria for Information Technology Security Evaluation - Part 1: Security assurance components, CCMB-2017-04-001, Version 3.1 Rev 5, April 2017*
	Common Criteria for Information Technology Security Evaluation - Part 2: Security assurance components, CCMB-2017-04-002, Version 3.1 Rev 5, April 2017
	Common Criteria for Information Technology Security Evaluation - Part 3: Security assurance components, CCMB-2017-04-003, Version 3.1 Rev 5, April 2017
	Common Criteria for Information Technology Security Evaluation – Evaluation methodology, CCMB-2017-04-0004, Version 3.1 Rev 5, April 2017
[TRM]	*Arm® Cortex®-M35P Processor Technical Reference Manual* (100883)
[IIM]	*Arm® Cortex®-M35P Processor Integration and Implementation Manual* (100886)
[UGRM]	*Arm® Cortex®-M35P Processor User Guide Reference Manual* (100887)
[PEN]	*Arm® Cortex®-M35P Product Errata Notice* (AT627-DC-11000)
[RN]	*Arm® Cortex®-M35P Processor Release Note* (PJDOC-466751330-5402)
[ArchSupp]	*Arm® Cortex®-M35P v8-M Architecture Supplement* (PJDOC-466751330-1229)
[Arm ARM]	*Arm®v8-M Architecture Reference Manual (*DDI0553A.i)

1.3. Terms and Abbreviations

This document uses the following terms and abbreviations:

Term or abbreviation	Meaning
AHB	AMBA High-performance Bus
Armv8-M	Armv8-M architecture described in the *Arm®v8-M Architecture Reference Manual*
BPU	Breakpoint Unit
C-AHB	Code AHB

CC	Common Criteria
CTI	Cross-Trigger Interface
CoreSight	Arm on-chip debug and trace components, that provide the infrastructure for monitoring, tracing, and debugging a complete system on chip.
D-AHB	Debug AHB
DAP	Debug Access Port
DMA	Direct Memory Access
DSP	Digital Signal Processing
DWT	Data Watchpoint and Trace
EAL	Evaluation Assurance Level
EPPB	External Private Peripheral Bus
ETM	Embedded Trace Macrocell
IC	Integrated Circuit
IDAU	Implementation Defined Attribution Unit
IRQ	Interrupt Request
ISR	Interrupt Service Routine
ITM	Instruction Trace Macrocell
FPU	Floating Point Unit
GPIO	General Purpose Input/Output
MPU	Memory Protection Unit
MTB	Micro Trace Buffer
NIC	Network Interconnect
NVIC	Nested Vectored Interrupt Controller
PC	Program Counter
PMSA	Protected Memory System Architecture
PP	Protection Profile
S-AHB	System AHB
SAR	Security Assurance Requirement
SAU	Security Attribution Unit
SCS	Secure Control Space
SE	Armv8-M Security Extension
SFR	Security Functional Requirement
SPD	Security Problem Definition
SPM	Security Policy Model

ST	Security Target
SWO	Serial Wire Output
TOE	Target of Evaluation
TPA	Trace Port Analyzer
TPIU	Trace Port Interface Unit
PPB	Private Peripheral Bus
WIC	Wake-up Interrupt Controller

1.4. Glossary

See the Arm glossary **http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html**.

1.5. Typographical conventions

This ST refers to [PP84] and [CC], as mentioned in **References** and therefore cites content directly copied from these references. In this ST, the original text copied from these references is typed as indicated here.

2 Introduction

2.1. ST reference

This ST is entitled *Arm® Cortex®-M35P r1p1 Lite Security Target*. The document version is 1.1 and dated 01/16/2020.

2.2. TOE reference

The TOE is identified as *Cortex-M35P r1p1*. It corresponds exactly to the components described in **Table 1 TOE physical scope**.

2.3. TOE overview

The TOE is the set of functionalities for a processor in a Security microcontroller IC. In this document, the TOE is also referred to as *the processor*. The intended environment for the TOE is the Security IC for smart card applications or similar services as identified and described in [PP84]. The TOE provides the functionality for software execution and controlling access to memory addresses in a Security IC.

The following figure shows a simplified general architecture of a Security IC microcontroller with the TOE processor part indicated and the major interfaces to the surrounding Security IC components.

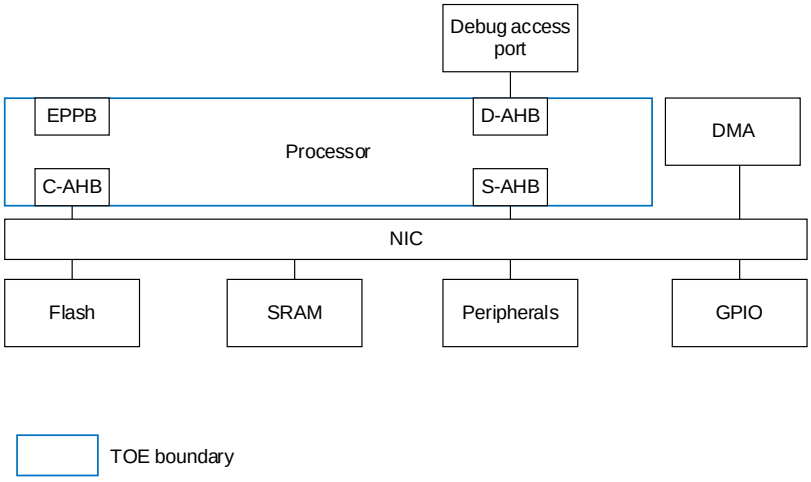


Figure 1 TOE component of a Security IC

The user of the TOE is the designer of a Security IC microcontroller product that integrates the TOE into their design for the microcontroller product. In this document, this user is referred to as the IC Designer. The user of the TOE is also the programmer of the Security IC dedicated software and the programmer of the Security IC embedded software that use the TOE programming interfaces consisting of the TOE instruction set and exception handling. It is the responsibility of the IC designer to instruct the programmer how to use the TOE.

The TOE is delivered as source code to be integrated by the IC Designer into the source code of their Security microcontroller product. To this end, the TOE has several interfaces that facilitate integration:

- Code AHB (C-AHB) interface.

    - The C-AHB interface is used for any instruction fetch and data access to the Code region of the Armv8-M memory map.
- System AHB (S-AHB) interface.
    - The S-AHB interface is used for any instruction fetch and data access to the memory-mapped SRAM, Peripheral, External RAM and External device, or Vendor_SYS regions of the Armv8-M memory map.
- External PPB (EPPB) interface.
    - The EPPB interface enables access to TOE external debug and trace components in a system connected to the processor.
- Debug AHB (D-AHB) interface.
    - The D-AHB slave interface allows a TOE external debugger access to registers, memory, and peripherals. The D-AHB interface provides debug access to the processor and the complete memory map.

See the **TOE logical scope** section for more information on these interfaces.

The TOE is compliant with the Armv8-M mainline architecture described in the *Arm®v8-M Architecture Reference Manual*. The Armv8-M mainline architecture is adaptable and enables a customer to tailor the functionality of the processor in their IC design from what are called architecture extensions. The following architecture extensions are available in the Cortex-M35P processor:

- The MPU Extension, which adds memory access control functionality.
- The Security Extension (SE), which adds security management. This extension can be referred to as Arm® TrustZone® for the TOE and requires the MPU Extension to be included.
- The floating-point (FPU) Extension.
- The Digital Signal Processing (DSP) Extension.
- The Debug Extension, which includes a debugger component that can be used for testing purposes during IC manufacturing.

The IC Designer chooses to include or exclude these extensions in their processor implementation using configuration parameters as part of the processor integration process, which is the first step they undertake after the delivery of the TOE to integrate the TOE in their IC design.

2.4. TOE identification

The TOE is the Cortex-M35P r1p1 product. Cortex-M35P r1p1 identifies the TOE including its components and guidance documentation listed in the **Physical scope** section. r1p1 is the version of the evaluated product where r is the major revision of the product (variant) and p is the minor revision of the product (revision). This version is identified in the *Arm® Cortex®-M35P Processor Release Note.* Any change in the TOE components or guidance documentation leads to a new version of the evaluated product, thus a new TOE.

2.5. TOE description

2.5.1 TOE logical scope

The following figure shows the TOE logical scope by identifying the logical components of the processor and the interfaces to the TOE environment.

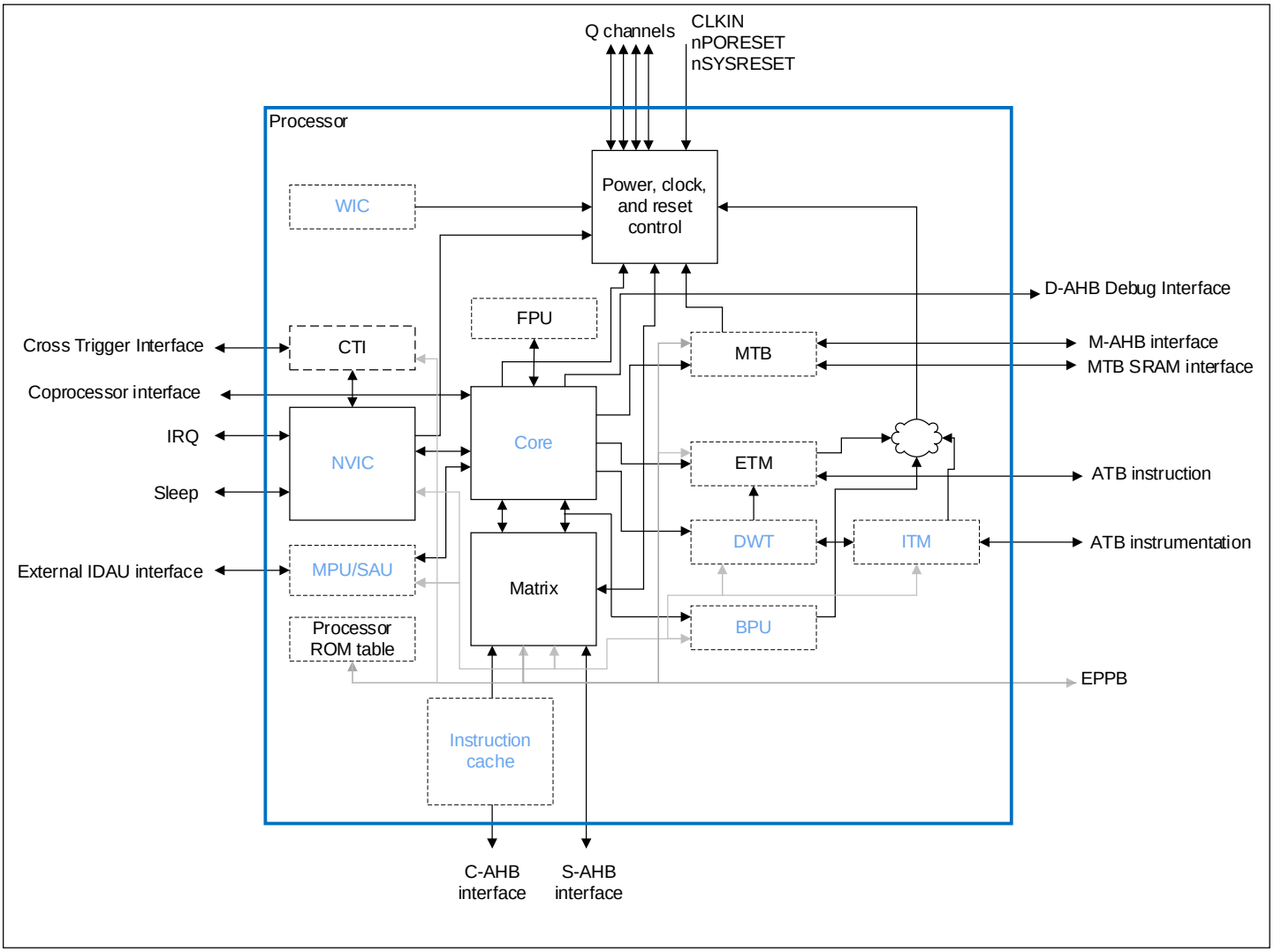


Figure 2 TOE logical scope

This figure shows the main interfaces identified in **Figure 1 TOE component of a Security IC** and details other interfaces. Components marked with dotted lines are optional as they are part of architecture extensions.

Although the MPU is part of the MPU extension and therefore can be optionally included or excluded during processor integration, it should always be included for a certified configuration.

Components in blue are configurable during processor integration. For example, the number of programmable memory regions in the MPU and SAU can be configured during processor integration.

The following subsections give an overview of the logical scope for each component.

Processor core

The processor core provides:

- Full support for the Armv8-M Security Extension.
- Harvard 32-bit AHB bus interfaces with vector fetch capability on the instruction side.
- 2/3 stage pipeline with early completion of common arithmetic instructions.
- Single-cycle branch latency.
- Limited dual-issue of common 16-bit instruction pairs.
- Single cycle 32×32-bit multiplier with two-cycle result latency for MAC operations.
- Integer divide unit with support for operand-dependent early termination.
- Support for interrupted continuable load and store multiple operations.
- Load and store operations both support precise errors.

Instruction cache

The instruction cache has the following features:

- 2-way set associative.
- 16-byte cache lines.
- Configurable size of 2KB, 4KB, 8KB, or 16KB.
- One-cycle penalty on uncacheable transactions.
- Read transactions are subject to caching and write transactions invalidate the cache.

Security attribution and memory protection (SAU/MPU)

The Cortex-M35P processor supports the Armv8‑M PMSA that provides programmable support for memory protection using software controllable regions.

Memory regions can be programmed to generate faults when they are inappropriately accessed by unprivileged software, reducing the scope of incorrectly written application code. The architecture includes fault status registers to allow an exception handler to determine the source of the fault and to apply corrective action or notify the system.

The Cortex-M35P processor also includes optional support for defining memory regions as Secure or Non-secure, as defined in the Armv8−M Security Extension, and protecting the regions from accesses with an inappropriate level of security.

Floating Point Unit (FPU)

The FPU provides:

- Instructions for single-precision (C programming language float type) data-processing operations.
- Instructions for double-precision (C double type) load and store operations.
- Combined multiply-add instructions for increased precision (Fused MAC).
- Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root.
- Hardware support for denormals and all IEEE Standard 754-2008 rounding modes.
- 32×32-bit single-precision registers or 16×64-bit double-precision registers.
- Lazy floating-point context save. Automated stacking of floating-point state is delayed until ISR attempts to execute a floating-point instruction. This reduces the latency to enter the ISR and removes floating-point context save for ISRs that do not use floating-point.

Nested Vectored Interrupt Controller (NVIC)

The NVIC is closely integrated with the core to achieve low-latency interrupt processing.

Functions of the NVIC include:

- External interrupts, configurable from 1 to 480 using a contiguous or non-contiguous mapping. This is configured at implementation.
- Configurable levels of interrupt priority from 8 to 256. This is configured at implementation.
- Dynamic reprioritisation of interrupts.
- Priority grouping. This enables selection of preempting interrupt levels and non-preempting interrupt levels.
- Support for tail-chaining and late arrival of interrupts. This enables back-to-back interrupt processing without the overhead of state saving and restoration between interrupts.
- Optional support for the Armv8‑M Security Extension. Secure interrupts can be prioritized above any Non-secure interrupt.

Wake-up Interrupt Controller (WIC)

The optional WIC supports the powering down of the core for power saving purposes. The WIC is responsible for registering pending exceptions and detecting wakeup conditions.

Power, clock, and reset control

The power, clock, and reset control is responsible for interfacing with an external PMU and generating the main clocks and resets for each power domain.

Cross Trigger Interface Unit (CTI)

The optional CTI enables the debug logic, MTB, and ETM to interact with each other and with other CoreSight components.

Embedded Trace Macrocell (ETM)

The optional ETM provides instruction-only capabilities when configured.

Micro Trace Buffer (MTB)

The MTB provides a simple low-cost execution trace solution for the Cortex-M35P processor.

Trace is written to an SRAM interface, and can be extracted using a dedicated AHB slave interface on the processor, MTB AHB. The MTB can be controlled by memory mapped registers in the PPB region or by events generated by the DWT or through the CTI.

Debug and trace additional components

Debug and trace components include:

- A configurable Breakpoint Unit (BPU) that implements a configurable number of comparators for breakpoints generation.
- A configurable Data Watchpoint and Trace unit (DWT) that implements watchpoints, data tracing, and system profiling.
- An Instruction Trace Macrocell (ITM) that creates a printf() style debug trace.
- A ROM table that allows debuggers to discover the list of debug components included in the Cortex-M35P processor.

Matrix

The matrix is a multilayer interconnect that routes the memory access requested by the core to the selected destination such as the C-AHB, S-AHB, EPPB, or internal PPB registers. Port selection is done according to the transaction address.

Interfaces

C-AHB and S-AHB

The C-AHB and S-AHB ports implement the AMBA5 AHB protocol and connect the core to external memories. These ports are physically driven by the matrix, and C-AHB can also be driven by the instruction cache if present. In a typical implementation, C-AHB connects to flash memories and S-AHB is dedicated to SRAMs.

External IDAU interfaces

The TOE includes support for an external attribution unit to allow the security level associated with a given address to be defined by the system. The output of this port is the address that need to be checked, and the input is the security information provided by the external attribution unit.

Sleep

Thanks to this interface, the TOE shares its power state with the external system. When the core enters a low-power state, it uses this port to communicate this information and the type of sleep it has entered (SLEEPING or DEEPSLEEP).

IRQ

This port collects all the external interrupts that are forwarded to the NVIC which is responsible for prioritizing and treating the interrupts. The TOE can support up to 480 external interrupts.

Coprocessor interface

The external Coprocessor interface allows the integration of tightly coupled hardware accelerators. The TOE accesses this interface using the architectural coprocessor instructions. This interface supports up to eight separate coprocessors. The system can configure which coprocessors are included in Secure and Non-secure state.

Q-Channels

A Q-Channel is a low-power interface. It supports the handshake mechanism that allow an external power controller to request the entrance of a TOE power domain into a quiescent state. Each TOE power domain has its own Q-channel interface. Each power domain uses the interface to share internal activity when it has stopped and then enters a low-power state.

Debug interface D-AHB

The debug interface is an AMBA5 AHB slave port. It allows an external debugger agent to connect to the internal TOE resources. This includes:

- The registers in the system.
- The memory-mapped devices.
- The internal core registers when the core is halted.
- The debug control registers even when reset is asserted.

External memory can also be accessed.

EPPB

The EPPB is a 32-bit AMBA4 APB interface designed for integration with the Debug and trace components and ROM tables. It is used for data only accesses to the memory region `0xE0040000-0xE00FFFFF` and it is not intended for general peripheral usage.

ATB instrumentation and ATB instruction

These two ports implement the AMBA4 ATB interface protocol which is a used by trace components to pass format-independent trace data through a CoreSight system. These ports are driven by the ITM and ETM respectively.

MTB SRAM interface and M-AHB interface

MTB SRAM interface is a dedicated port driven by the MTB which generates the execution trace to be written in the MTB SRAM. The M-AHB is an AMBA5 AHB slave port that allows reading the content of the SRAM passing through the MTB.

Cross Trigger Interface

The Cross Trigger Interface is a standard interface that enables the core debug logic and ETM and MTB to interact with each other and with additional CoreSight debug and trace components in the system external to the TOE.

Countermeasures

In addition, the TOE includes features that protect against information leakage and malfunction.

2.5.2 Physical scope

The following table lists the set of hardware, software, and document components that are delivered.

Component	Type	Version
Cortex-M35P Synthesizable Verilog	Verilog code (.v files)	AT627-MN-22110-r1p1-00rel0
Cortex-M35P Execution Test Bench	Testbench (C code)	AT627-MN-22010-r1p1-00rel0
Cortex-M35P Functional Test Source	Testbench (C code)	AT627-VE-70006-r1p1-00rel0
Cortex-M35P RAM Integration Test Bench	Testbench (C code)	AT627-MN-70002-r1p1-00rel0
Arm® Cortex®-M35P Product Errata Notice	Document (.pdf)	AT627-DC-11000-r1p1-00rel0
Arm® Cortex®-M35P Release Note	Document (.pdf)	AT627-DC-06003-r1p1-00rel0
Arm® Cortex®-M35P Processor Technical Reference Manual	Document (.pdf)	AT627-DA-03001-r1p1-00rel0
Arm® Cortex®-M35P Processor Integration and Implementation Manual	Document (.pdf)	AT627-DC-70047-r1p1-00rel0
Arm® Cortex®-M35P Processor User Guide Reference Manual	Document (.pdf)	AT627-DA-03005-r1p1-00rel0
Arm® Cortex®-M35P v8-M Architecture Supplement	Document (.pdf)	AT627-DC-50000-r1p1-00rel0
Arm® Cortex®-M35P r1p1 Security Guidance	Document (.pdf)	PJDOC-466751330-8802 3.1

Table 1 TOE physical scope

Note:

All documents are restricted by nature. Access to these documents depends on your agreement with Arm.

2.6. TOE lifecycle and delivery

This ST claims a part of Protection Profile [PP84]. The following figure shows the lifecycle for composite products based on a Security IC as described in [PP84]. The development of the TOE covers a part of Phase 2. The delivery of the TOE takes place halfway Phase 2.

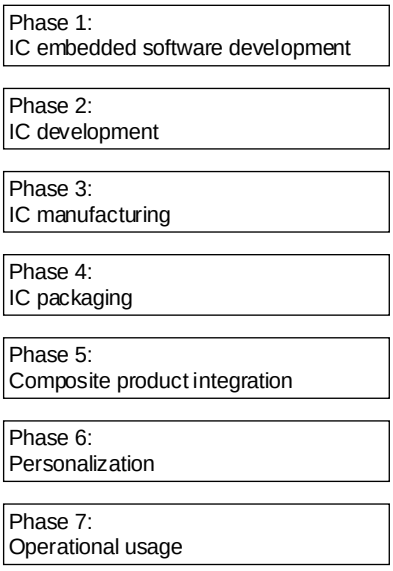


Figure 3 Lifecycle according to [PP84]

After delivery of the TOE source code to customers, the IC designer starts integrating the TOE into their IC design in the middle of Phase 2.

2.7. Configuration

The Cortex-M35P processor is configurable. As part of the integration process, the IC designer can decide to include or exclude optional extensions identified in the **TOE overview** section. The IC designer can also choose the size of configurable countermeasures (see the **TOE summary specification** chapter for details on configuration of countermeasures). After configuration, the identity of the TOE is determined by the source code version and the set of configuration parameters used. This identity is assessed during the sign-off process, which is based on log files that result from the configuration process.

3 Conformance claims

3.1. CC conformance claims

This document claims to be conformant to the CC version 3.1. Furthermore, it claims to be conformant to the CC Part 2 and conformant to the CC Part 3.

This document has been built with the *Common Criteria for Information Technology Security Evaluation; Version 3.1* which comprises:

- *Common Criteria, Part 1: Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, Version 3.1, Revision 5, April 2017.*
- *Common Criteria, Part 2: Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Components, Version 3.1, Revision 5, April 2017.*
- *Common Criteria, Part 3: Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Components, Version 3.1, Revision 5, April 2017.*
- *Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017.*

3.2. PP claim

This ST does not claim conformance to a PP. This ST claims a subset of the *Security Problem Definition and the Security Requirements of the IC Platform Protection Profile [PP84].*

The purpose of this document is to enable the Designer of Security IC products to certify their product from a composite evaluation, in which the certification results of this TOE can be reused. To this end, the [PP84] objectives that are not implemented in the TOE have been claimed as objectives for the environment of the TOE.

3.3. Package claim

This ST claims conformance to the assurance package EAL6 augmented with ASE_TSS.2 and ALC_FLR.1.

3.4. Conformance claim rationale

The purpose of this ST is to provide a platform certificate to a composite evaluation for a Security microcontroller claiming [PP84]. The EAL of [PP84] is EAL4 augmented with the assurance components ALC_DVS.2 and AVA_VAN.5. The assurance claim of this ST includes all assurance components of [PP84].

By claiming EAL6, this ST can provide a base component for Security IC evaluations claiming conformance to [PP84], up to and including EAL6.

By claiming ASE_TSS.2, this ST also explains as part of the summary specification how the TOE protects itself against bypass and tampering.

By claiming ALC_FLR.1, this ST also explains how Arm protects the TOE and corporate information.

4 Security problem definition

4.1. Core SPD and additional SPD

The **PP claim** section explains that the purpose of this ST is to provide a base part to a composite evaluation for a Security IC that claims conformance to [PP84]. The intention of this ST is also to help the Security IC designer claim the security for their IC product, in line with the choices that the configuration process allows. To this end, the SPD in this ST has a core part which defines the security problem for all possible processor configurations resulting from the configuration process, and additional SPDs for the services that the extensions provide. The following figure shows the scope of the different parts in the SPD.

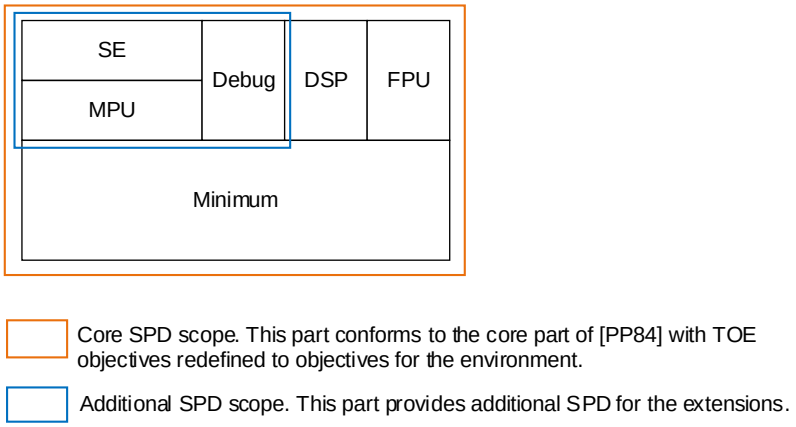


Figure 4 Core and additional SPD

This figure shows that the SE depends on both the services of the MPU Extension and the minimum functionality. The other extensions are only dependent on the minimum functionality.

The core part of the SPD is according to the core part of [PP84], through which some TOE objectives have been redefined to objectives for the environment. The SPD description for the extensions add threats, organisational security policies, assumptions, and objectives specific to the extensions. The ST for the user Security IC claims compliance to [PP84] and additionally claims threats, organisational security policies, assumptions, and objectives for the extensions that remain in the design of the Security IC after TOE configuration.

There is a distinction between additional SPD and optional extensions. Additional SPDs are SPDs in addition to the [PP84]. These additional SPDs map to functional extensions of the TOE architecture that can be optionally included or excluded during the customer integration process. However, to have a meaningful TOE configuration as basis for certification the MPU extension must be included as a minimum during the integration process.

After delivery of the TOE, the customer ST for the user Security IC will claim compliance to [PP84] and additionally will claim SPD (threats, organisational security policies, assumptions, and objectives) for the extensions that remain in the design of the Security IC after TOE integration.

4.2. Description of assets

The assets to be protected are the end-user software and data in the external memories, which are known in [PP84] as IC dedicated software and embedded software. The TOE shall protect the confidentiality and integrity of the end-user data and software when it is transported to the TOE and when being processed by the core.

4.3. Core SPD

The core of the SPDs describes the security problem to be solved for all TOE configurations after the user configuration process. This core SPD is according to a subset of the [PP84] SPD.

4.3.1 Core threats

Because the TOE is regarded a component of a Security IC, all threats defined in *Section 3.2* of [PP84] are regarded applicable to the TOE, except T.RND. [PP84] considers T.RND as a threat for a security service required in a Security microcontroller. Because this security service is not provided by the TOE, it is regarded not applicable for the TOE.

The following table shows the threats of [PP84] that are applicable to the TOE.

Threat name	Threat definition
T.Leak-Inherent	Inherent Information Leakage
T.Phys-Probing	Physical Probing
T.Malfunction	Malfunction due to Environmental Stress
T.Phys-Manipulation	Physical Manipulation
T.Leak-Forced	Forced Information Leakage
T.Abuse-Func	Abuse of Functionality

Table 2 Threats defined in [PP84] applicable to the TOE

4.3.2 Core organisational security policies

The following table shows the organisational security policy that is defined in *Section 3.3* of [PP84] which is also applicable to the TOE.

Threat name	Threat definition
P.Process-TOE	Identification during TOE Development and Production An accurate identification is established for the TOE. This requires that each instantiation of the TOE carries this unique identification.

Table 3 Organisational security policy

4.3.3 Core assumptions

The following table shows the assumptions defined in *Section 3.4* of [PP84] that are also applicable to the TOE.

Assumption name	Assumption definition
A.Process-Sec-IC	Protection during Packaging, Finishing, and Personalisation
A.Resp-Appl	Treatment of user data of the Composite TOE

Table 4 Assumptions defined in [PP84] applicable to the TOE

4.4. Additional SPD

The additional SPD adds threats, organisational security policies, assumptions, and objectives specific to the TOE extensions.

4.4.1 Additional threats

The following table shows the additional Threat specific to the Debug extension.

Threat name	Threat definition
T.Debug-abuse	Misuse of the TOE Debug functionality An attacker might misuse the TOE debug interface to get access to end-user data in the memories, or registers in the core, that should not be accessible according to the TOE access control policies.

Table 5 Threat specific to the Debug Extension

There are no additional threats specific to the other extensions.

4.4.2 Additional organisational security policies

The following table shows additional organisational security policy specific to the MPU Extension and Security Extension (SE).

Threat name	Threat definition
P.Mem-Access	Memory region-based access control The TOE must enable the IC dedicated software and the end-user embedded software to manage and control access to regions in memory.

Table 6 Organisational security policy specific for the MPU extension and Security Extension (SE)

4.4.3 Additional assumptions

There are no additional Assumptions for the TOE extensions.

5 Security objectives

In line with the SPD description in the **Security problem definition** section, the security objectives are described as core objectives and objectives for the extensions.

5.1. Core security objectives

Core objectives are the objectives that apply to any configuration resulting from the user configuration process. Core objectives are objectives from [PP84].

5.1.1 Core security objectives for the environment

The objectives for the environment in [PP84] are also claimed in this ST. In addition, a part of the objectives for the TOE in [PP84] cannot be the responsibility of the TOE and therefore become objectives for the environment in this ST.

The following table identifies the objectives for the environment of the TOE that have been copied from [PP84].

Objective name	Objective definition
OE.Process-Sec-IC	Protection during composite product manufacturing
OE.Resp-Appl	Treatment of user data of the Composite TOE

Table 7 Objectives for the environment from [PP84]

The following table identifies the objectives for the environment of the TOE that have resulted from redefining and renaming objectives in [PP84].

Objective name	Objective definition
OE.Phys-Probing	Protection against Physical Probing
OE.Leak-Forced	Protection against Forced Information Leakage
OE.Abuse-Func	Protection against Abuse of Functionality
OE.Identification	TOE Identification
OE.Phys-Manipulation	Protection against Physical Manipulation

Table 8 Redefined and renamed objectives for the environment from [PP84]

5.1.2 Core security objectives for the TOE

The following table shows the objectives that have been copied from [PP84] that are also claimed by the TOE.

Objective name	Objective definition
O.Leak-Inherent	Protection against Inherent Information Leakage
O.Malfunction	Protection against Malfunctions

Table 9 Objectives for the TOE copied from [PP84]

5.1.3 Core security objective rationale

The following table shows how the objectives address the assumptions, threats, and organisational security policies for the core objectives.

Assumption, threat, or organisational security policy	Security objective	Notes
T.Leak-Inherent	O.Leak-Inherent	-
T.Phys-Probing	OE.Phys-Probing	T.Phys-Probing has been redirected to OE.Phys-Probing to become an objective for the environment.
T.Malfunction	O.Malfunction	-
T.Phys-Manipulation	OE.Phys-Manipulation	T. Phys-Manipulation has been redirected to OE. Phys-Manipulation to become an objective for the environment.
T.Leak-Forced	OE.Leak-Forced	T.Leak-Forced has been redirected to OE.Leak-Forced to become an objective for the environment.
T.Abuse-Func	OE.Abuse-Func	T.Abuse-Func has been redirected to OE.Abuse-Func to become an objective for the environment.
A.Resp-Appl	OE.Resp-Appl	-
A.Process-Sec-IC	OE.Process-Sec-IC	-
P.Process-TOE	OE.Identification	P.Process-TOE has been redirected to OE.Identification to become an objective for the environment.

Table 10 Security objectives rationale for the core security objectives

The following table shows the assumptions, threats, or organisational security policies identified in [PP84] that have not been included in this ST, and gives the reason why.

Assumption, threat, or organisational security policy in [PP84]	Reason for not including in this ST
T.RND	T.RND in [PP84] is regarded a threat for a random number security service in a security microcontroller. Because this security service is not provided by the TOE, it is regarded not applicable for the TOE.

Table 11 Assumptions, threats, or organisational security policies not included in the ST

5.2. Additional security objectives

Additional security objectives are objectives specific to the Armv8-M extensions.

5.2.1 Additional security objectives for the environment

The following table identifies the objectives for the environment that are specific to the TOE Debug extension.

Objective name	Objective definition
OE.Auth-Debug	Debug authorisation Before an external debugger is allowed access to processor resources and memory from the debug interface, security procedures must be used to verify the authorized use.

Table 12 Objectives for the environment specific to the TOE Debug extension

Note:

Access to internal registers of the core from an external debug component is possible in debug mode. The processor enters debug mode when the external debug component halts the processor. The debug component can halt the processor by writing to a control register. Otherwise, the external debug component can access the memory of the register in the PPB space. A combination of signals in an implementation-defined authentication interface to the processor determines whether halting is prohibited or allowed from the debug interface. These external authentication signals enable the processor to decide whether the debug component can halt the processor or not.
Arm provide several debug components that can interface to the processor Debug Interface, but these are not included in the TOE delivery.

The software-controlled Debug Authentication Control Register can override the hardware signals in the authentication interface. This provides flexibility in implementing an authentication function in the IC design.

There are no additional objectives for the environment specific to the other extensions.

5.2.2 Additional security objectives for the TOE

The following table shows the additional security objectives for the MPU Extension.

Objective name	Objective definition
O.Mem-Access	Memory region-based Access Control The TOE must provide the IC dedicated software and the end-user embedded software with the capability to define restricted access to memory regions. The TOE must enforce the access of software to these memory regions depending on access privileges.

Table 13 TOE objective specific for the TOE MPU Extension

The following table shows the additional security objectives for the Security Extension (SE).

Objective name	Objective definition
O.Oper-Access	Operation-based Access Control The TOE must provide the IC dedicated software and the end-user embedded software with the capability to define types of software code in memory regions. The TOE must enforce the access of running software to software code in memory regions depending on the requested operation between them.

Table 14 TOE Objectives specific for the TOE Security Extension (SE)

There are no additional TOE objectives for the other extensions.

5.2.3 Additional security objective rationale

The following table shows how the objectives for the Armv8-M extensions address the assumptions, threats, and organisational security policies specific for the Armv8-M extensions.

Assumption, threat, or organisational security policy	Security objective	Notes
T.Debug-abuse	OE.Auth-Debug	The objective for the environment OE.Auth-Debug covers the threat. The OE.Auth-Debug instructs the user to implement an authentication mechanism to the use of an external debug component connected to the TOE debug interface, thereby covering the T.Debug-abuse threat.
P.Mem-Access	O.Mem-Access	The objectives O.Mem-Access and O.Oper-access both cover the organisational security policy O.Mem-Access that states that IC dedicated software and end-user embedded software must be able to manage and control access to regions in memory. The O.Mem-Access does this from attributes given to memory regions and privileges from software running in the different processor modes. The O.Oper-access does this from attributes given to memory regions and the security state of the entire processor.
	O.Oper-access	

Table 15 Security objectives rationale for the additional security objectives

6 Extended components definition

This ST has no definitions for extended SFRs.

7 Security requirements

7.1. Typographical conventions

Security requirements include SFRs and SARs. Operations in SFRs and SARs are described in ***bold italic font***.

7.2. Core SFRs

The core SFRs implement the TOE core objectives. The following table shows the SFRs from the [PP84] that implement the TOE core objectives.

Name	Title	Addressing	Description
FDP_ITT.1	Basic internal transfer protection	Leakage	See section 173 in [PP84].
FDP_IFC.1	Subset information flow control	Leakage	See section 175 in [PP84].
FPT_ITT.1	Basic internal TSF data transfer protection	Leakage	See section 174 in [PP84].
FRU_FLT.2	Limited Fault Tolerance	Perturbation	See section 151 in [PP84].
FPT_FLS.1	Failure with preservation of secure state	Perturbation	See section 152 in [PP84].

Table 16 SFRs from [PP84] that comprise the TOE core SFRs

7.3. SFRs for the MPU Extension

The following table shows the SFRs that implement the TOE objectives for the MPU Extension.

Name	Title	Addressing	Description
FDP_ACC.2.1/MPU	Complete access control - MPU	Memory access violation	See **Table 18 Security requirements for FDP_ACC.2/MPU**.
FDP_ACC.2.2/MPU			
FDP_ACF.1.1/MPU	Security based access control - MPU		See **Table 19 Security requirements for FDP_ACF.1/MPU**.
FDP_ACF.1.2/MPU			
FDP_ACF.1.3/MPU			
FDP_ACF.1.4/MPU			
FMT_MSA.3.1/MPU	Static attribute initialisation - MPU	Correct operation	See **Table 26 Security requirements for FMT_MSA.3/SE.**
FMT_MSA.3.2/MPU			
FMT_MSA.1.1/MPU	Management of security attributes - MPU		See **Table 27 Security requirements for FMT_MSA.1/SE**.
FMT_SMF.1.1/MPU	Specification of management functions - MPU		See **Table 28 Security requirements for FMT_SMF.1/SE**.

Table 17 SFRs from [PP84]

In addition to SFRs, according to [PP84], the TOE provides source code for access control services that enable privileged software in the Security IC to separate and manage IC memory resources among multiple applications.

7.3.1 Memory access protection

The MPU Extension allows privileged software to define memory regions and assigns access permissions to them. If the access permission that are assigned to the memory regions are violated, then the TOE fault management is triggered by memory accesses.

Access control definition

The scope of access control services is defined by the Access control security function policy in the FDP_ACC SFR in terms of:

- The subjects under the control of the policy.
- The objects under the control of the policy.
- The operations between the controlled objects and subjects.

Subjects

The TOE policy for memory access protection is based on the following subject:

Software

Software includes the instructions being executed by the processor core. As part of software execution, software requires access to memory addresses. Depending on the operating mode of the processor, software can be privileged software or unprivileged software.

Objects

The TOE policy for memory access protection is based on the following object:

Memory address

Memory addresses give access to locations in memory where code or data is stored, or device registers can be accessed that enable control over peripherals. Memory addresses are identified by a 32-bit word which makes the locations of code and registers accessible through a bus interface.

Operations

The TOE policy for memory access protection is based on the following operations:

- **Read** Read data from a location in memory or from a device register.
- **Write** Write data to a location in memory or to a device register.
- **Execute** Execute an instruction that is fetched from memory.

Security attributes

The SFR that defines the access control functions, FDP_ACF, describes the rules that control the access control policy and the attributes that are used in the rules.

The TOE policy for memory access control uses the following security attributes:

Operating mode

Operating mode is an attribute of the processor when executing software. The operating modes are:

- **Privileged** Software that is executed in the privileged mode of the processor is also referred to as privileged software.
- **Unprivileged** Software that is executed in the unprivileged mode of the processor is also referred to as unprivileged software.

The TOE processor also has a halt mode operating mode, but in this mode the MPU access control is not used.

MPU region

The range of memory addresses in the TOE is divided into regions. Each region is identified with a number and each region has a base address and a limit address that identify the range of consecutive addresses in the region. MPU regions have the following attributes:

- **Region-enabled** The MPU region is enabled.
- **Region disabled** The MPU region is disabled.

If the SE is included, that is, if the processor implementation is configured with the Security Extension, then the MPU Extension has two sets of MPU regions. Depending on the security state of the processor, the MPU uses either the Secure or the Non-secure set. When MPU regions are accessed to be modified, they are addressed as Secure MPU regions or Non-secure MPU regions.

MPU control

This attribute controls whether the MPU is enabled and whether the MPU can use the default memory map for access control. The MPU control has the following attributes:

- **MPU-enabled** The MPU is enabled.
- **MPU-disabled** The MPU is disabled. When the MPU is disabled, the regions are not used for access control resolution. Instead, the default memory map is used.
- **Background-enabled** The default memory map regions can be used as background regions by privileged software only. If background usage is enabled, then the MPU access control policy uses the default memory map if the requested address cannot be mapped in any of the regions.
- **Background-disabled** The default memory map regions cannot be used as background regions.

Access permissions

Define allowable access to the range of addresses in a region. The possible values are:

- **0b00** Read/write by privileged code only.
- **0b01** Read/write by any privilege level.
- **0b10** Read-only by privileged code only.

- **0b11** Read-only by any privilege level.

eXecute Never (XN)

Defines whether code can be executed in this region. The possible values are:

- **0** Execution only permitted if read permitted.
- **1** Execution not permitted.

Default memory map

In addition to the set of MPU regions, the TOE has a default memory map with fixed regions and fixed attribute values for eXecute Never, shareability, and cacheability. This default memory map is used for access control resolution when either the entire MPU is disabled or when the access control policy has been allowed to use the default memory map when a requested address cannot be mapped in any of the regions. Such access control resolution is only performed in the case of privileged software.

Note:

Memories also have attributes for shareability and cacheability that respectively can identify if the memory can be shared among different masters and the type of cache behavior for optimal cache performance. These attributes however are not used in the TOE access control policies.

7.3.2 Security functional requirements

The following table shows the requirements for Complete access control (FDP_ACC.2) that the TOE shall meet.

FDP_ACC.2/MPU	Complete access control
Hierarchical to	FDP_ACC.1 Subset access control
Dependencies:	FDP_ACF.1 Security attribute based access control
FDP_ACC.2.1/MPU	The TSF shall enforce the ***MPU access control policy*** on ***all software and all memory addresses and all operations*** among subjects and objects covered by the SFP.
FDP_ACC.2.2/MPU	The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

Table 18 Security requirements for FDP_ACC.2/MPU

Note:

The access control policy shall be enforced by implementing an MPU. Before a respective memory address is accessed by the processor core, the MPU checks whether the respective operation is allowed.

The following table shows the requirements for Security attribute based access control (FDP_ACF.1) that the TOE shall meet.

FDP_ACF.1/MPU	Security attribute based access control
Hierarchical to	No other components

FDP_ACF.1/MPU	Security attribute based access control
Dependencies	• FDP_ACC.1 Subset access control • FMT_MSA.3 Static attribute initialisation
FDP_ACF.1.1/MPU	The TSF shall enforce the ***MPU access control policy*** to objects based on the following: ***All subjects and objects and the attributes Operating mode, MPU Region, MPU control, Access permissions, and eXecute Never.***
FDP_ACF.1.2/MPU	The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: ***A read or write operation to a memory address is allowed:*** ▪ ***If the address is within the address range of one the MPU regions and if the current operating mode and requested operation are allowed in the Access permissions of that region.*** ***An execute operation to a memory address is allowed:*** ▪ ***If the address is within the address range of one the MPU regions, and if the current operating mode and read operation are allowed in the Access permissions of that region, and if execution is permitted in the eXecute Never of that region.***
FDP_ACF.1.3/MPU	The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: ▪ ***When the MPU control is MPU-disabled and background-enabled, and none of the regions is region-enabled, then the region attributes of the default memory map are used to check access to addresses only if the current operating mode is privileged.*** ▪ ***When the MPU control is MPU-disabled, the region attributes of the default memory map are used to check access to addresses for both privileged and unprivileged software.***
FDP_ACF.1.4/MPU	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: ▪ ***All operations are denied when the requested address matches more than one MPU region.***

Table 19 Security requirements for FDP_ACF.1/MPU

The following table shows the requirement for Static attribute initialisation (FMT_MSA.3) that the TOE shall meet.

FMT_MSA.3/MPU	Static attribute initialisation
Hierarchical to	No other components
Dependencies:	• FMT_MSA.1 Management of security attributes • FMT_SMR.1 Security roles
FMT_MSA.3.1/MPU	The TSF shall enforce the ***MPU access control policy*** to provide ***restrictive*** default values for security attributes that are used to enforce the SFP.
FMT_MSA.3.2/MPU	The TSF shall allow ***privileged software*** to specify alternative initial values to override the default values when an object or information is created.

Table 20 Security requirements for FMT_MSA.3/MPU

Note:

Restrictive means that the reset value of the processor core is privileged and the reset value of the MPU is MPU-disabled and background-disabled, leaving it to privileged software to first program the MPU regions from access provided by the default memory map.

The following table shows the requirement for Management of security attributes (FMT_MSA.1) that the TOE shall meet.

FMT_MSA.1/MPU	Management of security attributes
Hierarchical to	No other components
Dependencies	• FDP_ACC.1 Subset access control or FDP_IFC.1 Subset information flow control • FMT_SMR.1 Security roles • FMT_SMF.1 Specification of management functions
FMT_MSA.1.1/MPU	The TSF shall enforce the **MPU *access control policy*** to restrict the ability to ***modify*** the security attributes • ***Secure MPU region, MPU control, Access permissions, and eXecute Never.*** **to** • ***Privileged software in the Secure state of the processor.*** **and** • ***Non-secure MPU region, MPU control, Access permissions, and eXecute Never.*** **to** • ***Privileged software in the Non-secure state and the Secure state of the processor.***

Table 21 Security requirements for FMT_MSA.1/MPU

Note:

In combination with the Security Extension (SE), the MPU Extension has two sets of MPU regions, one Secure and one Non-secure, and the processor has a Secure state and a Non-secure state. If the SE is excluded from the user implementation, then the Secure MPU region defaults to Non-secure MPU region and the processor security state defaults to Non-secure state. The user ST then claims the third operation in the FMT_MSA.1/MPU as ***MPU region, MPU control, Access permissions and eXecute Never*** **to** ***Privileged software***.

The following table shows the requirement for Specification of Management Functions (FMT_SMF.1) that the TOE shall meet.

FMT_SMF.1/MPU	Static attribute initialisation
Hierarchical to	No other components
Dependencies	No dependencies
FMT_SMF.1.1/MPU	The TSF shall be capable of performing the following management function: ▪ ***Modification of the security attributes from accessing TOE registers by privileged software.***

Table 22 Security requirements for FMT_SMF.1/MPU

7.4. SFRs for the Security Extension

The following table shows the SFRs that implement the objectives for the Armv8-M Security Extension (SE).

Name	Title	Addressing	Description
FDP_ACC.2.1/SE	Complete access control - SE	Memory access violation	See **Table 24 Security requirements for FDP_ACC.2/SE.**
FDP_ACC.2.2/SE			
FDP_ACF.1.1/SE	Security based access control - SE		See **Table 25 Security requirements for FDP_ACF.1/SE.**
FDP_ACF.1.2/SE			
FDP_ACF.1.3/SE			
FDP_ACF.1.4/SE			
FMT_MSA.3.1/SE	Static attribute initialisation - SE	Correct operation	See **Table 26 Security requirements for FMT_MSA.3/SE.**
FMT_MSA.3.2/SE			
FMT_MSA.1.1/SE	Management of security attributes - SE		See **Table 27 Security requirements for FMT_MSA.1/SE.**
FMT_SMF.1.1/SE	Specification of management functions - SE		See **Table 28 Security requirements for FMT_SMF.1/SE.**

Table 23 SFRs from [PP84]

7.4.1 SE access control

The SE enables the system and software to be partitioned into Secure and Normal worlds. Secure software can access both Secure and Non-secure memories and resources, whereas Normal software can only access Non-secure memories and resources. These security states are separated from the existing privileged and unprivileged modes, enabling both a privileged and unprivileged mode in both Secure and Non-secure states.

The following figure shows the security states and modes.

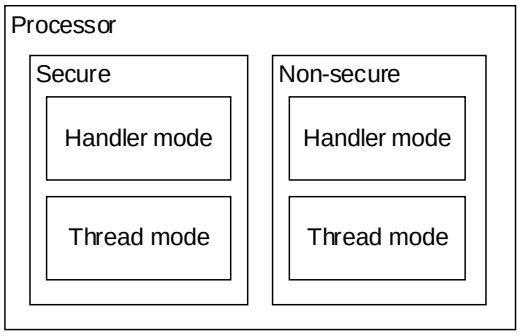


Figure 5 Lifecycle according to [PP84]

In applications that are based on Security microcontrollers supporting the SE, the software components that are critical to the security of the system can be placed in the Secure world. These critical components typically include:

- A Secure boot loader.
- Secret keys.
- Flash programming support.
- High value assets.

The remaining applications are placed in the Normal world.

Secure (Trusted) and Non-secure (Non-trusted) software can work together, but Non-secure applications cannot access Secure resources directly. Instead, any access to Secure resources can go through APIs provided by Secure software (and these APIs can implement authentications to decide if the access to the Secure service is permitted).

7.4.2 Access control definition

The scope of access control services is defined by the access control security function policy in the FDP_ACC SFR in terms of:

- The subjects under the control of the policy.
- The objects under the control of the policy.
- The operations between the controlled objects and subjects.

Subjects

The TOE policy for SE access control is based on the following subjects:

Software

Software includes the instructions being executed by the processor core. In the context of the SE access control, software is the program code running in the current security state of the processor, which can be Secure or Non-secure. Software is also identified as Secure software or Non-secure software.

Debug component

A debug component that is connected to the processor can halt the processor and inspect the core registers and external memory.

Objects

The TOE policy for SE access control is based on the following objects:

Code in memory

Software code in memory is the collection of instructions and data stored in address regions of the memory. These can be Secure code or Non-secure code depending on the given value of their security attributes. Depending on the security attributes given to software in memory, software in memory is also referred to as Secure memory or Non-secure memory.

Peripherals

Peripherals are the collection of registers accessible from memory addresses that interface to peripherals. Depending on the given values of device attributes, peripherals can be Secure peripherals or Non-Secure peripherals.

Operations

The TOE policy for SE access control policy is based on the following operations:

Instruction execution

This operation fetches the next instruction for execution by the core. To check if the instruction execution is allowed, SE access control checks if the address of the instruction is of the memory type that the SE requires.

Data access

This operation is the read or write operation to an address as part of an instruction execution. Because the architecture of the TOE core complies to a load/store architecture, the target address for the read or write operation is in one of the general-purpose registers. The SE access control checks if the read or write operation to that target address is allowed according to the type of memory on the address.

Cross call

This operation is performed by a limited set of instructions that can call software in the other security world. The SE checks if the instruction is performed from the allowed security world from an allowed address. After the operation, the security state of the core has been changed.

In a simplified view, the SE allows Non-secure software to call or jump to software in Non-secure memory only, or perform data access operations on Non-secure memory only. Secure software can call or access both Secure and Non-secure memory. State transition operations from Secure state to Non-secure memory are only allowed to limited instructions. For state transitions from Non-secure state, a special restriction is imposed to ensure that only valid Secure API entry points can be used for calling from the Normal world to Secure code, or for returning to the original state.

The following instructions are available for state transition handling:

- **Secure Gateway (SG)**
  This instruction is used in switching from Non-secure to Secure state. It must be the first instruction of the Secure entry point.
- **Branch with exchange to Non-secure state (BXNS)**
  This instruction is used by Secure software to branch or return to Non-secure program.
- **Branch with link and exchange to Non-secure state (BLXNS)**
  This instruction is used by Secure software to call Non-secure functions.

The following figure shows the security state transitions:

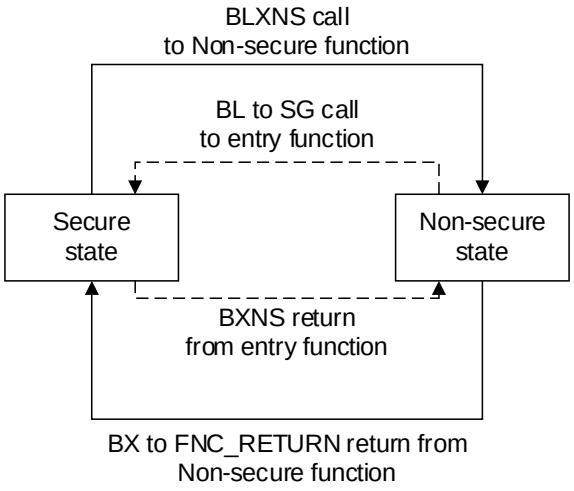


Figure 6 Security state transitions

Attributes

The SE memory access control policy uses the following security attribute:

Operating mode

For the MPU access control policy, the operating mode is an attribute of software being executed by the processor and can be privileged and non-privileged. For the SE access control policy, the halt mode of the processor is also relevant.

Halt mode

In the halt mode of the processor, software execution from memory is halted, enabling a debug component to control registers in the SCS of the core. The processor halt mode is also referred to as the processor debug mode.

Processor security state

Processor security state can be considered as an attribute of software. It defines the current security state of the processor when executing software instructions. In a simplified view, software can be regarded Secure or Non-secure based on the security state of the processor.

The processor security state can be:

Non-secure

Program code executed in the Non-secure state of the processor is referred to as Non-secure software.

Secure

Program code executed in the Secure state of the processor is referred to as Secure software.

If the processor is in Secure state, then it must fetch instructions from Secure memory.

SE control

This security attribute controls whether the SE access control policy is enabled or not. It has the following attributes:

- **SE-enabled** The SE access control is enabled.
- **SE-disabled** The SE access control is disabled.

SE region

With the SE, the 4GB memory space is partitioned into Non-secure and Secure memory regions. There is only one set of SE regions across the Secure and Non-secure states of the processor. Each SE region has a base address and an ending address that define the range of consecutive addresses in the region. Each SE region has a set of attributes describing the type of memory in the region. SE regions have the following attributes:

- **Region-enabled** The SE region is enabled.
- **Region-disabled** The SE region is disabled.

SE region memory types

The memories in the SE regions can be:

Non-secure (NS memory)

Non-secure addresses are used for memory and peripherals accessible by all software that is running on the device.

Secure (S memory)

Secure addresses are used for memory and peripherals accessible only by Secure software or masters.

Non-secure Callable (NSC)

NSC is a special type of Secure memory location. This memory type is the only type which the processor permits to hold an SG instruction that allows software to transition from Non-secure to Secure state[1].

1 The inclusion of NSC locations avoids a need for managing accidental inclusion of SG instructions or data in normal Secure memory. Typically, NSC memory is used for library components starting with a jump table.

7.4.3 Security functional requirements

The following table shows the requirements for Complete access control (FDP_ACC.2) that the TOE shall meet when the SE is included.

FDP_ACC.2/SE	Complete access control
Hierarchical to	FDP_ACC.1 Subset access control
Dependencies	FDP_ACF.1 Security attribute based access control
FDP_ACC.2.1/SE	The TSF shall enforce ***the SE access control policy*** on ***all software accessing all code and peripherals in memory*** and all operations among subjects and objects covered by the SFP.
FDP_ACC.2.2/SE	The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

Table 24 Security requirements for FDP_ACC.2/SE

Note:

The subjects of all software in the SE access control policy mean all program code executed by the processor either in Secure state or Non-secure state. The object of all code and peripherals means all instructions, data, and peripherals in memory marked as either Secure or Non-secure.

The following table shows the requirements for Security attribute based access control (FDP_ACF.1) that the TOE shall meet.

FDP_ACF.1/SE	Security attribute based access control
Hierarchical to	No other components
Dependencies	• FDP_ACC.1 Subset access control • FMT_MSA.3 Static attribute
FDP_ACF.1.1/SE	The TSF shall enforce ***the SE access control policy*** to objects based on the following: **All subjects, all objects, and the attributes processor security state and SE memory type**.

FDP_ACF.1/SE	Security attribute based access control
FDP_ACF.1.2/SE	The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: • ***The SE allows data access operations with the processor in Secure state on all memory types.*** • ***The SE allows data access operations with the processor in Non-secure state only if the memory of the SE region that contains the target address is Non-secure.*** • ***The SE allows instruction execution operations with the processor in Secure state only if the memory of the SE region that contains the target address is Secure.*** • ***The SE allows instruction execution operations with the processor in Non-Secure state only if the memory of the SE region that contains the target address is Non-secure.*** • ***The SE allows cross-call operations from the processor in Non-secure state only if the address of the called function is in a Non-secure Callable memory SE region and if the first instruction of the called function is an SG instruction.*** • ***The SE allows cross-call return operations from the processor in Secure state only to the BLXNS instruction with the RETURN code in the LR register.*** • ***The SE cross-call return operations from the processor in Non-secure state only to the BX instruction with the FNC_RETURN code in the LR register.*** ***The SE allows access to Secure memory from a debug component if the processor is in debug halt mode and if the external debugger requests Secure access.***
FDP_ACF.1.3/SE	The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: • ***A target address that matches multiple SE regions is marked as Secure memory regardless of the memory types specified by the regions that matched the target address.*** • ***If SE control is SE-disabled, then the required memory type of the target address is marked as either Secure or Non-secure memory, depending on the choice made by the IC Designer during TOE integration.*** • ***If none of the SE regions marked region-enabled contain the target address, or if multiple SE regions contain the target address, then the required memory type of the target address is set to Secure memory.*** • ***Operations to addresses allowed by the SE must be also allowed by the MPU access control policy to be performed.***
FDP_ACF.1.4/SE	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: • ***None.***

Table 25 Security requirements for FDP_ACF.1/SE

The following table shows the requirements for Static attribute initialisation (FMT_MSA.3) that the TOE shall meet.

FMT_MSA.3/SE	Static attribute initialisation
Hierarchical to	No other components
Dependencies	• FDP_ACC.1 Subset access control • FMT_MSA.3 Static attribute
FMT_MSA.3.1/SE	The TSF shall enforce the ***SE access control policy*** to provide ***restrictive*** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3/SE	Static attribute initialisation
FMT_MSA.3.2/SE	The TSF shall allow ***None*** to specify alternative initial values to override the default values when an object or information is created.

Table 26 Security requirements for FMT_MSA.3/SE

Note:
Restrictive means that the processor resets in Secure state.

The following table shows the requirements for Management of security attributes (FMT_MSA.1) that the TOE shall meet.

FMT_MSA.1/SE	Management of security attributes
Hierarchical to	No other components
Dependencies	• FDP_ACC.1 Subset access control or FDP_IFC.1 Subset information flow control • FMT_SMR.1 Security roles • FMT_SMF.1 Specification of Management Functions
FMT_MSA.1.1/SE	The TSF shall enforce the ***SE access control policy*** to restrict the ability to ***modify*** the security attributes in ***SE regions*** to ***privileged software in processor Secure state***.

Table 27 Security requirements for FMT_MSA.1/SE

The following table shows the requirements for Specification of Management Functions (FMT_SMF.1) that the TOE shall meet.

FMT_SMF.1/SE	Management of security attributes
Hierarchical to	No other components
Dependencies	No dependencies
FMT_SMF.1.1/SE	The TSF shall be capable of performing the following management function: • ***Modification of the SE security attributes from accessing TOE registers by privileged software in processor Secure states.***

Table 28 Security requirements for FMT_SMF.1/SE

7.5. Security assurance requirements

The ST is evaluated according to the Security Target evaluation, Class ASE. The TOE claims EAL6 augmented by ASE_TSS.2 and ALC_FLR.1.

The following table shows the set of SARs claimed by the TOE, indicating the origin of each SAR.

Class	SAR
Class ADV: Development	Architectural design (ADV_ARC.1)
	Security Policy Model (ADV_SPM.1)

Class	SAR
	Functional Specification (ADV_FSP.5)
	Implementation Representation (ADV_IMP.2)
	TSF Internals (ADV_INT.3)
	TOE Design (ADV_TDS.5)
Class AGD: Guidance documents	Operational User Guidance (AGD_OPE.1)
	Preparative procedures (AGD_PRE.1)
Class ALC: Life-cycle support	CM Capabilities (ALC_CMC.5)
	CM Scope (ALC_CMS.5)
	Delivery (ALC_DEL.1)
	Development Security (ALC_DVS.2)
	Flaw remediation (ALC_FLR.1)
	Lifecycle Definition (ALC_LCD.1)
	Tools and Techniques (ALC_TAT.3)
Class ASE: Security Target evaluation	Conformance claims (ASE_CCL.1)
	Extended components definition (ASE_ECD.1)
	ST introduction (ASE_INT.1)
	Security objectives (ASE_OBJ.2)
	Derived security requirements (ASE_REQ.2)
	Security problem definition (ASE_SPD.1)
	TOE summary specification (ASE_TSS.2)
Class AVA: Vulnerability assessment	Advanced methodical vulnerability analysis (AVA_VAN.5)
Class ATE: Tests	Coverage (ATE_COV.3)
	Depth (ATE_DPT.3)
	Functional Tests (ATE_FUN.2)
	Independent Testing (ATE_IND.2)

Table 29 SARs claimed by the TOE

The following table shows the claimed security policy as specified by the assurance requirement for ADV_SPM.1.

ADV_SPM.1	Formal TOE security policy model
ADV_SPM.1.1D	The developer shall provide a formal security policy model for ***the memory access control policy and the operation access control policy.***
ADV_SPM.1.2D	For each policy covered by the formal security policy model, the model shall identify the relevant portions of the statement of SFRs that make up that policy.

ADV_SPM.1	Formal TOE security policy model
ADV_SPM.1.3D	The developer shall provide a formal proof of correspondence between the model and any formal functional specification.
ADV_SPM.1.4D	The developer shall provide a demonstration of correspondence between the model and the functional specification.

Table 30 Claimed security policy for ADV_SPM.1

7.5.1 Refinements of the TOE assurance requirements

Because this ST claims all assurance requirements from [PP84] and because the objective of this ST is to provide a base platform for certification of [PP84] compliant security ICs, this ST also claims all SAR refinements from [PP84]. The following table lists these SARs.

SAR	Refinement	Description
ALC_DEL	188 in [PP84]	For delivery of the TOE to the "Composite Product Manufacturer" as consumer, all the external interfaces of the TOE Manufacturer have to be taken into account. This ST redefines this refinement to: For delivery of the TOE to the "IC Designer as consumer", all the external interfaces of the composite TOE designer have to be taken into account.
ALC_DVS	194 in [PP84]	"TOE design and implementation" must be understood as comprising all material and information related to the development and production of the TOE.
ALC_CMS	199 in [PP84]	This refinement in [PP84] is out of scope for the TOE because it relates to consumer software that can be part of manufacturing and delivery.
ALC_CMC	205 in [PP84]	This refinement in [PP84] is out of scope for the TOE because it refers to the CMS refinement.
	206 in [PP84]	This refinement in [PP84] is out of scope for the TOE because it refers to tracking of production batches for wafers or dies.
ADV_ARC	209 in [PP84]	The Security Architecture description of the TSF initialisation process shall include the procedures to establish full functionality after power-up, state transitions from the secure state as required by FPT_FLS.1, and any state transitions of power save modes if provided by the TOE.
	210 in [PP84]	This refinement in [PP84] is out of scope for the TOE because it relates to test features used in wafer testing.
ADV_FSP	215 in [PP84]	This refinement refers to test software delivered but not available in the operational phase. This refinement is regarded out of scope for the TOE.
	216 in [PP84]	This refinement refers to features that do not provide functionality but nevertheless contribute to SFRs. This refinement is regarded out of scope for the TOE.
	217 in [PP84]	The Functional Specification is expected to refer to mechanisms.
	218 in [PP84]	This refinement refers to operating conditions. This refinement is regarded out of scope for the TOE.
ADV_IMP	223 in [PP84]	It must be checked that the provided implementation representation is complete and sufficient to ensure that analysis activities are not curtailed due to lack of information.

SAR	Refinement	Description
ATE_COV	226 in [PP84]	This refinement specifies that the TOE must be tested under different operating conditions within the specified ranges. This refinement is out of scope for the TOE.
	227 in [PP84]	This refinement relates to physical testing. This refinement is out of scope for the TOE.
AGD_OPE	233 in [PP84]	[…] the role of the developer of the Security IC Embedded Software is the main focus of the guidance […]. This ST redefines this refinement to: [...] the role of the IC Designer of the Security IC Embedded Software is the main focus of the guidance [...].
	234 in [PP84]	This refinement relates to requirements concerting embedded software. This requirement is regarded out of scope for the TOE.
	235 in [PP84]	Guidance documents must not contain security relevant details which are not necessary for the usage or administration of the security functionality of the TOE.
AGD_PRE	239 in [PP84]	The Family AGD_PRE addresses the activities of the delivery acceptance procedures.
	240 in [PP84]	This refinement refers to configuration in Phase 2 or Phase 7. This refinement is out of scope for the TOE.
	241 in [PP84]	This refinement refers to downloading of embedded software. This refinement is out of scope for the TOE.
AVA_VAN	245 in [PP84]	The vulnerability analysis shall include a justification for the rating of information on the TOE available to the attacker.

Table 31 Refinements on the TOE assurance requirements

7.6. Security requirements rationale

The security requirements rationale identifies the modifications and additions made to the rationale presented in [PP84].

7.6.1 Rationale for the security functional requirements

The following table shows how the SFRs are combined to meet the security objectives.

Objective	TOE SFRs
O.Leak-Inherent	• FPT_ITT.1 - Basic internal TSF data transfer protection • FDP_ITT.1 - Basic internal transfer protection • FDP_IFC.1 - Subset information flow control
O.Malfunction	• FRU_FLT.2 - Limited Fault Tolerance • FPT_FLS.1 – Failure with preservation of secure state

Objective	TOE SFRs
O.Oper-Access	• FDP_ACC.2/SE - Complete access control • FDP_ACF.1/SE - Security attribute based access control • FMT_MSA.3/SE - Static attribute initialisation • FMT_MSA.1/SE - Management of security attributes • FMT_SMF.1/SE - Specification of Management Functions
O.Mem-Access	• FDP_ACC.2/MPU - Complete access control • FDP_ACF.1/MPU - Security attribute based access control • FMT_MSA.3/MPU - Static attribute initialisation • FMT_MSA.1/MPU - Management of security attributes • FMT_SMF.1/MPU - Specification of Management Functions

Table 32 Mapping of SFRs to the TOE objectives

Justification for Protection against Inherent Information Leakage (O.Leak.Inherent)

The following table shows the justification related to the security objective Protection against Inherent Information Leakage (O.Leak-Inherent).

SFR	Justification
FDP_ITT.1 FDP_IFC.1	The refinement of FDP_ITT.1 together with the policy statement in FDP_IFC.1 require the prevention of secret user data disclosure, either when transmitted between separate parts of the TOE or while being processed. This includes that attackers cannot reveal such data by measurements of emanations, power consumption, or other behavior of the TOE either when the data is transmitted between separate parts of the TOE or when it is processed.

Table 33 Justification for O.Leak-Inherent

Justification for Protection against Malfunctions (O.Malfunction)

The following table shows the justification related to the security objective Protection against Malfunctions (O.Malfunction).

SFR	Justification
FRU_FLT.2 FPT_FLS.1	The definition of this objective covers the situation where malfunction of the TOE might be caused by direct manipulation of the TOE: • The first case is covered by FRU_FLT.2 because it states that the TOE operates correctly under normal (tolerated) conditions. • The second case is covered by FPT_FLS.1 because it states that a secure state is preserved in this case. The functions implementing FRU_FLT.2 and FPT_FLS.1 must work independently so that their operation cannot be affected by the Security IC Embedded Software (refer to the refinement). Therefore, there is no possible instance of conditions under O.Malfunction that is not covered.

Table 34 Justification for O.Malfunction

Justification for Operation based Memory Access Control (O.Oper-Access)

The following table shows the justification related to the security objective Operation based Memory Access Control (O.Oper-Access).

SFR	Justification
FDP_ACC.2/SE	The SFR with the respective SFP require the implementation of an area-based memory access control, which is a requirement from O.Oper-Access. Therefore FDP_ACC.2/SE, with its SFP, is suitable to meet the security objective O.Oper-Access.
FDP_ACF.1/SE	The SFR: • Allows the TSF to enforce access to objects within the respective SFP based on security attributes. • Defines these attributes and defines the rules based on these attributes that enable explicit decisions. Therefore FDP_ACF.1/SE, with its reference to the SFP and rules based on security attributes, is suitable to meet the security objective O.Oper-Access.
FMT_MSA.3/SE	The SFR requires that the TOE provides default values for the security attributes used in the SFP. Because the TOE is a hardware platform, these default values are generated by the reset procedure. Therefore, FMT_MSA.3/SE is suitable to meet the security objective O.Oper-Access.
FMT_MSA.1/SE	The SFR requires that authorized users can manage TSF attributes. It ensures that the access control attributes required by O.Oper-Access can be realized from attributes that can be managed from using the functions provided by the TOE. Being a hardware platform, the security attributes used in the SFP are accessible from register bits. Therefore, FMT_MSA.1/SE is suitable to meet the security objective O.Oper-Access.
FMT_SMF.1/SE	The SFR is used for the specification of the management functions to be provided by the TOE as required by O.Oper-Access. Being a hardware platform, the TOE allows the management of the security attributes by making the registers accessible to software to enable modification. Therefore, FMT_SMF.1/SE is suitable to meet the security objective O.Oper-Access.

Table 35 Justification for O.Oper-Access

Justification for Area based Memory Access Control (O.Mem-Access)

The following table shows the justification related to the security objective Area based Memory Access Control (O.Mem-Access).

SFR	Justification
FDP_ACC.2/MPU	The SFR with the respective SFP require the implementation of an area-based memory access control, which is a requirement from O.Mem-Access. Therefore FDP_ACC.2/MPU, with its SFP, is suitable to meet the security objective O.Mem-Access.

SFR	Justification
FDP_ACF.1/MPU	The SFR: • Allows the TSF to enforce access to objects within the respective SFP based on security attributes. • Defines these attributes and defines the rules based on these attributes that enable explicit decisions. Therefore FDP_ACF.1/MPU, with its reference to the SFP and rules based on security attributes, is suitable to meet the security objective O.Mem-Access.
FMT_MSA.3/MPU	The SFR requires that the TOE provides default values for the security attributes used in the SFP. Because the TOE is a hardware platform, these default values are generated by the reset procedure. Therefore, FMT_MSA.3/MPU is suitable to meet the security objective O.Mem-Access.
FMT_MSA.1/MPU	The SFR requires that authorized users can manage TSF attributes. It ensures that the access control attributes required by O.Mem-Access can be realized from attributes that can be managed from using the functions provided by the TOE. Being a hardware platform, the security attributes used in the SFP are accessible from register bits. Therefore, FMT_MSA.1/MPU is suitable to meet the security objective O.Mem-Access.
FMT_SMF.1/MPU	The SFR is used for the specification of the management functions to be provided by the TOE as required by O.Mem-Access. Being a hardware platform, the TOE allows the management of the security attributes by making the registers accessible to software to enable modification. Therefore, FMT_SMF.1/MPU is suitable to meet the security objective O.Mem-Access.

Table 36 Justification for O.Mem-Access

7.6.2 Dependencies of security functional requirements

The following table shows the SFRs defined in this ST, their dependencies, and whether they are satisfied by other security requirements defined in this ST.

SFR	Dependencies	Fulfilled by security requirements
FDP_ACC.2/MPU	FDP_ACF.1	Yes
FDP_ACF.1/MPU	FDP_ACC.1	Yes, fulfilled by FDP_ACC.2/MPU which is hierarchically higher
	FMT_MSA.3	Yes
FMT_MSA.3/MPU	FMT_MSA.1	Yes
	FMT_SMR.1	See footnote[2]
FMT.MSA.1/MPU	FDP_ACC.1 or FDP_IFC.1	Yes

[2] The dependency for FMT_SMR.1 introduced by the two components FMT_MSA.1/MPU and FMT_MSA.3/MPU is satisfied because the access control specified for the intended TOE is not role-based but enforced for each subject. Therefore, there is no need to identify roles in form of FMT_SMR.1 SFR.

SFR	Dependencies	Fulfilled by security requirements
	FMT_SMR.1	See footnote[2]
	FMT_SMF.1	Yes
FMT_SMF.1/MPU	None	No dependency
FDP_ACC.2/SE	FDP_ACF.1	Yes
FDP_ACF.1/SE	FDP_ACC.1	Yes, fulfilled by FDP_ACC.2/SE which is hierarchically higher
	FMT_MSA.3	Yes
FMT_MSA.3/SE	FMT_MSA.1	Yes
	FMT_SMR.1	See footnote[3]
FMT.MSA.1/SE	FDP_ACC.1 or FDP_IFC.1	Yes
	FMT_SMR.1	See footnote[3]
	FMT_SMF.1	Yes
FMT_SMF.1/SE	None	No dependency
FDP_ITT.1	FDP_ACC.1 or FDP_IFC.1	Yes
FDP_IFC.1	FDP_IFF.1	See footnote[4]
FPT_ITT.1	None	No dependency
FRU_FLT.2	FPT_FLS.1	Yes
FPT_FLS.1	None	No dependency

Table 37 Mapping of SFRs to TOE objectives

7.6.3 Rationale for the assurance requirements

The assurance level EAL6 was chosen to facilitate that the TOE can provide a platform certificate for Security IC evaluations at the level of EAL6. The requirement for ASE_TSS.2 provides transparency to users about the TOE features that provide protection against tampering, interference, and bypass. The requirement ALC_FLR.1 provides assurance to the users that Arm has policies and procedures to track and correct flaws, and to distribute the flaw information and corrections.

7.6.4 Dependencies of security assurance requirements

The TOE claims EAL6 augmented by ASE_TSS.2 and ALC_FLR.1. All dependencies of the assurance components in the EAL6 are met by the EAL6 assurance level definition. The augmented ASE_TSS.2 is dependent on assurance components that are either part of

[3] The dependency for FMT_SMR.1 introduced by the two components FMT_MSA.1/SE and FMT_MSA.3/SE is satisfied because the access control specified for the intended TOE is not role-based but enforced for each subject. Therefore, there is no need to identify roles in form of FMT_SMR.1 SFR.

[4] The rationale follows the explanation given in section 274 of [PP84].

EAL6 or for which EAL6 contains hierarchically higher components. The augmented ALC_FLR.1 is not dependent on any assurance component.

7.6.5 Security requirements are internally consistent

The core SFRs in this ST are based on the Security Problem Definition copied from the certified [PP84] with some objectives redefined to become objectives for the environment. Because [PP84] is internally consistent, the core part of this ST is for the same reasons also consistent.

Two objectives for the TOE have been added to the SPD to implement an organisational security policy that must allow embedded software to control access to memory and operations. The SFRs that implement these objectives provide services to embedded software that do not conflict with any of the [PP84] SFRs.

One of the additional objectives is an objective for the environment that ensures that the Debug extension is only used in an authenticated manner. This objective for the environment does not conflict with any of the [PP84] objectives.

The SARs in this ST augment the assurance requirements of [PP84] in most families. Because [PP84] is internally consistent for its SFRs and SARs, this ST is for the same reasons also consistent.

8 TOE summary specification

8.1. Security functions

The following sections describe how the TOE satisfies the SFRs in the form of Security Functions (SFs). The SFs that implement core SFRs are:

- SF.Leak.Protect – Leakage protection.
- SF.Malfunction – Malfunction protection.

The SFs that implement additional SFRs are:

- SF.MPU – MPU access control.
- SF.SE – SE access control.

8.1.1 SF.Leak.Prot - Leakage protection

This SF implements the following SFRs:

- FDP_ITT.1.
- FDP_IFC.1.
- FPT_ITT.1.

This SF provides a service to the IC dedicated software and end-user embedded software that helps prevent leakage of asset information from side channel analysis.

The functional behavior for SF.Leak.Prot can be implemented from:

- Random instruction insertion.
- Uniform timing.
- Data masking.
- Polarity.
- Register renaming.
- Instruction cache random invalidate.
- Instruction cache address scrambling.

The TOE is configurable, and the features that implement SF.Leak.Prot described in this section can be included or excluded, as described in the *Arm® Cortex®-M35P Processor Implementation and Integration Manual*. If the user does not include any of the features listed in this section, then the TOE does not provide the claimed security functionality. Therefore, if the user wishes to use this security functionality, then they must enable at least one of the features listed in their configuration.

8.1.2 SF.Malfunction - Malfunction protection

This SF protects the integrity of the TSF functionality of the processor and implements the FRU_FLT.2 and FPT_FLS.1 SFRs. Several features in the processor that can detect parity errors in registers or buses implement this SF. On detection of parity errors, the processor faults to its Secure reset state.

The TOE is configurable, and the parity feature described in this section is optional and can be enabled or disabled for various groups of registers or buses, as described in the *Arm® Cortex®-M35P Processor Implementation and Integration Manual*. If the user does not include parity for any registers or buses in their configuration, then the TOE does not provide the claimed security functionality.

Therefore, if the user wishes to use this security functionality, then they must enable parity for the registers or buses that require protection in their configuration.

8.1.3 SF.MPU - MPU access control

The functional behavior for SF.MPU is implemented by the source code provided in the MPU Extension of the TOE. This extension provides the source code for implementing the following SFRs:

- FDP_ACC.2/MPU.
- FDP_ACF.1/MPU.
- FMT_MSA.3/MPU.
- FMT_MSA.1/MPU.
- FMT_SMF.1/MPU.

For a certified processor in the customer IC design, the MPU extension must be included during the customer integration process.

When included in the IC design, the SF.MPU functionality is mainly provided by the MPU component added to the Cortex-M35P architecture. The MPU implements a large set of registers that allow control of the MPU from software. The control register in the MPU controls whether the MPU is enabled or not, and whether the default memory map must be used as background regions in the MPU control policy. The MPU also implements the logic that executes the MPU access control policy, and the MPU has logic that interfaces the MPU to the processor operating mode in the core. Therefore, the MPU implements the FDP_ACC.2/MPU, FDP_ACF.1/MPU, and FMT_MSA.3/MPU SFRs.

The core logic implements the FMT_SMF.1/MPU SFR that controls that only privileged software can access the MPU region registers from the default memory map in the SCS of the core through the PPB.

A set of registers in the MPU implement the MPU regions for the MPU access control policy. These registers implement the FMT_MSA.1 SFR, enabling user software to control the MPU attributes for address limits and access privileges. The region registers also have an enable bit that determines if they participate in the access control policy. The IC Designer determines the number of MPU regions that can participate in the implemented access control policy of the Security IC when the TOE is integrated in the IC design. When used in the enforcement of the access control policy, the region registers are only indirectly addressed by the MPU. This creates flexibly in allowing the IC Designer to decide on a variable number of regions to implement and it also facilitates the extension of the MPU with a second set of MPU regions for SE functionality, without changing the MPU logic for enforcement of the access control enforcement.

8.1.4 SF.SE – SE access control

The functional behavior for SF.SE is implemented by the optional source code provided in the MPU Extension and the SE of the TOE. Together these extensions provide the source code that implements the following SFRs:

- FDP_ACC.2/SE.
- FDP_ACF.1/SE.
- FMT_MSA.3/SE.
- FMT_MSA.1/SE.
- FMT_SMF.1/SE.

The source code in the SE adds a security state and extra instructions to the core, and several processor resources are doubled and become banked between the Secure and Non-secure state of the processor. In the core, for instance, the SP register is banked between Secure and Non-secure state. This enables separation between Secure and Non-secure stacks. When a register is banked in this way, there is a distinct instance of the register in Secure state and another distinct instance of the register in Non-secure state.

Depending on the processor state, the registers that belong to the Secure state or to the Non-secure state are used for software execution in the core. Special behavior is also added to the core to implement the logic for Cross-call operations in core instructions.

The added SE behavior in the core instruction set implements some of the rules in the FDP_ACF.1 SFR.
From the SE source code, a second set of MPU regions is added to the MPU that enable the MPU to conduct memory access control based on the security state.
This part of the source code implements the security checking of the SE access control policy, thus implementing the FDP_ACC.2/SE, FDP_ACF.1/SE, and FMT_MSA.3/SE SFRs.

The SAU and the IDAU are important components in the SE source code. The SAU implements the registers for the SAU regions that enable software to program the attributes used in the SE access control policy. Together these region registers implement the FMT_MSA.1/SE SFR. Programming the registers for the SAU regions is only allowed to privileged software. Privileged software can access these registers from absolute addressing them in the SCS through the PPB in the default memory map. Therefore, the core implements the FMT_SMF.1/SE SFR.

The number of SAU regions used by the SE access control policy is determined by the IC Designer when integrating the TOE in their design. The SAU derives the security attributes for a memory address. To do so, it might use the data from the external IDAU.
The purpose of the SAU is to return the security state attributed to an address from the SE region registers. The SE access control policy can also use attributes from an external non-TOE component. The IDAU is an optional external component that the IC Designer might want to implement as a configurable (non-programmable) default memory map, which can be overridden with SAU regions for some parts of the memory when needed.

The following figure shows how the SAU and IDAU work in the processor.

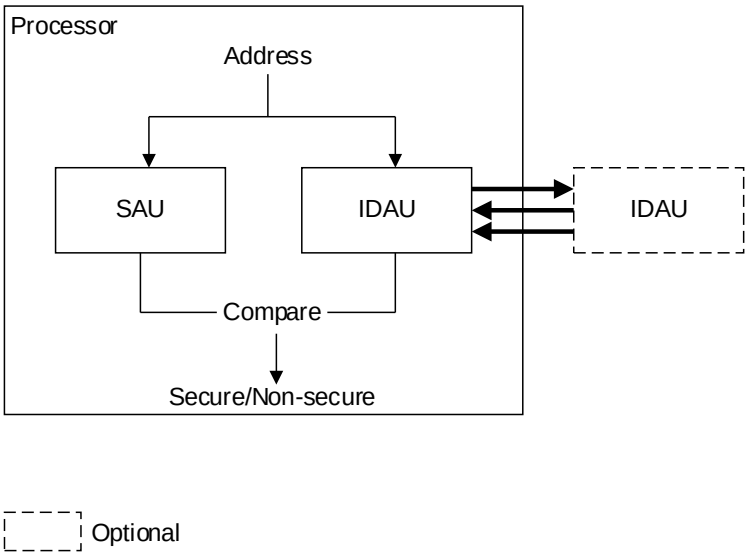


Figure 7 SAU and IDAU

The SAU in the optional SE and the optional IDAU in the user design define three levels of memory security attribution:

- Non-secure (NS).
- Secure and Non-secure callable (NSC).

- Secure and not Non-secure callable[5] (S).

The attribution information from the SAU is used for an address unless the IDAU specifies attributes with a higher security. In this case, the IDAU attributes override the SAU attributes.

Note:

For instruction fetches, the address range 0xF0000000-0xFFFFFFFF is always seen as Secure.

Moreover, the following address ranges are exempt of SAU/IDAU checking, which means that memory security is reported as NS-Req and no region information is supplied:

- 0xE0000000-0xEFFFFFFF (full range, in case of instruction fetch).
- 0xE0000000-0xE0002FFF (ITM, DWT, and FPU, exempt regions given by the IDAU).
- 0xE000E000-0xE000EFFF (SCS, exempt region given by the IDAU).
- 0xE002E000-0xE002EFFF (SCS Non-secure alias, exempt region given by the IDAU).
- 0xE0030000-0xE0030FFF (instruction cache, exempt region given by the IDAU).
- 0xE0040000-0xE0041FFF (TPIU and ETM, exempt regions given by the IDAU).
- 0xE0042000-0xE0043FFF (CTI and MTB, exempt regions given by the IDAU).
- 0xE00FF000-0xE00FFFFF (ROM table, exempt region given by the IDAU).

8.1.5 SF.SE uses SF.MPU

The following figure shows the sequence of address checking from SF.SE to SF.MPU.

In this figure, NS-Req defines the state where the processor requests that a memory access is performed. For a data access, NS-Req is the current security state of the processor. For an instruction fetch, NS-Req is the security state of the target address.

[5] Secure memory that cannot be called from Non-secure software. See the *Arm®v8-M Architecture Reference Manual.*

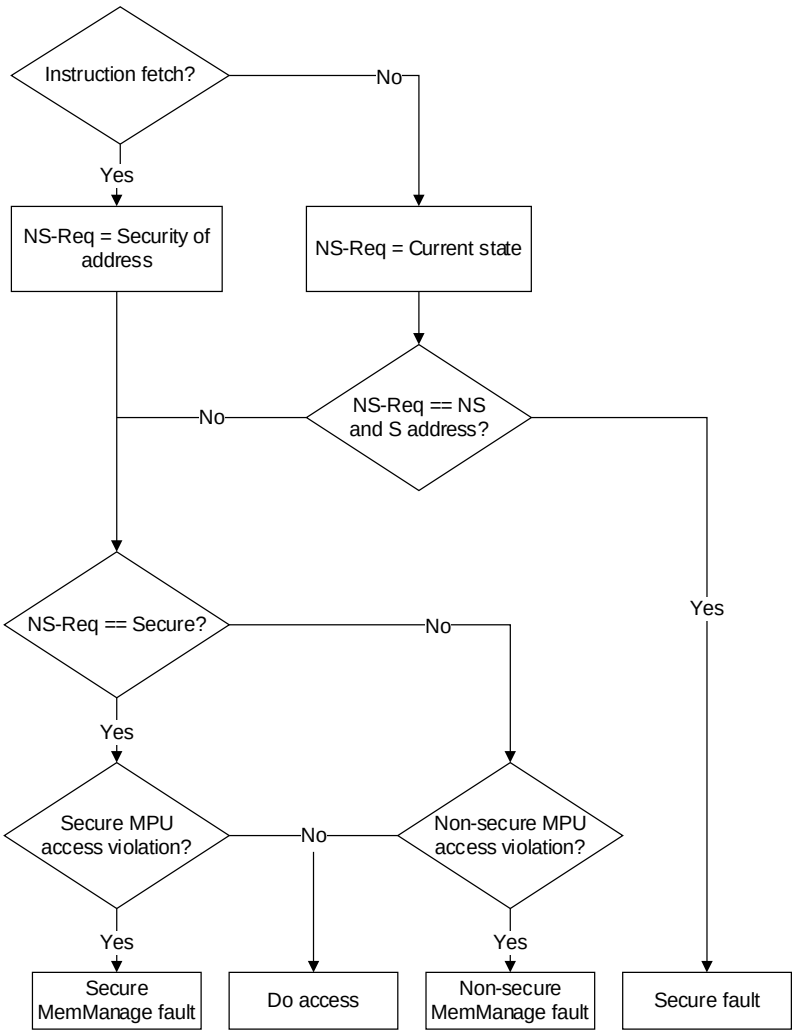


Figure 8 Security attribution and MPU check sequence

The following example shows that, in case of a data access, address checking might result in a Secure fault without involvement of the MPU:

```
IF not Instruction call          / This is a Data access operation  /
       THEN
         NS-Req = current state:
         IF NS-Req == Non-secure AND data address == Secure
                                    / IF current state is Non-secure and
                                    / address for data access is Secure
         THEN                       / THEN trigger fault
            Secure fault
            EXIT
```

FDP_ACF.1/SE defines that for data access, non-privileged software can only access data in Non-secure memory. In all other cases, the MPU is always involved in address checking.

The following example shows the case of an instruction fetch:

```
IF instruction fetch             / in case of Instruction fetch
       THEN                              / SE checks the PC address
          NS-Req = Secure OR Non-secure
                                         /from the SE region content
       IF NS-Req == Secure       / if NS-Req dictates secure memory
       THEN                              / Secure MPU regions must be used
              Current MPU = Secure MPU regions
              IF access allowed          / if NX allowed on address in region
              THEN
                 Do access        / fetch and execute instruction
              ELSE
                 Secure MemManage fault
       ELSE                              / else NS-Req dictates Non-secure
                                         / Non-secure MPU regions must be used
              Current MPU = Non-secure MPU regions
              IF access allowed          / if NX allowed on address in region
              THEN
                 Do access        / fetch and execute instruction
              ELSE
                 Non-secure MemManage fault.
```

Data access operations and state transition operations also follow this sequence, whereby data access operations do not involve the XN attribute in the MPU regions. For state transition operations, NS-Req requires the other state compared to the current operating state for the SE to check.

8.2. Protection against tampering and bypass

This ST claims the assurance requirement ASE_TSS.2, that is supposed to give high-level information on how the TOE protects itself against interference and logical tampering, and how the TOE protects itself against bypass.

The TOE functionality complies to the Armv8-M mainline architecture. In this mainline Armv8-M architecture, the SM.Log-Tamper-Protect and SM.Bypass-Protect mechanisms protect against logical tampering and bypass.

8.2.1 SM.Log-Tamper-Protect

The Armv8-M architecture has been designed to recognize conditions at the TOE interfaces that might compromise the TOE security functionality. In response to erroneous inputs, the TOE generates exceptions that can be handled by software from an interrupt vector or by external hardware from a signal or an external interface. This makes the TOE robust against the many different programming mistakes in software and errors from external hardware.

Examples of Armv8-M features that protect against logical tampering include:

Fault recognition and handling

In the Armv8-M architecture faults can be recognized to be:

- A HardFault.
- A MemManage fault.
- A BusFault.
- A UsageFault.

When the Security Extension is included, an additional SecureFault is defined. The architecture specifies the order of priority among the exceptions, considering that priority order of MemManage, BusFault, UsageFault, and SecureFault is configurable.

Fault handling is done in software from handlers activated from vector tables. Fault Status Registers (xFSR) are available for each type of fault with bits allocated to specific faults that allow handler programmers to identify the cause of a fault.

With this Security Mechanism, the processor guarantees that any unexpected behavior is treated with the intended handler.

Hardware-controlled priority escalation to HardFault and lockup state

The Armv8-M architecture defines that when a processor is executing an exception handler which has a certain priority level, and its execution creates itself an exception of equal or lower priority, then the processor automatically escalates the derived exception priority to a HardFault priority level. In this way, the processor can avoid any deadlock and the derived exception can be correctly served.

On the other hand, when escalation to HardFault handler is not possible because of the current execution priority, the processor automatically enters a state called lockup state. For example, this can happen when the HardFault exception is already active when the escalation is required. In lockup state, the processor mainly stops fetching and executing instructions, and signals its state to the system by asserting the **LOCKUP** output signal. Exiting from lockup state is only possible with a reset (Cold or Warm) or a preemption by another priority.

This feature guarantees that any unexpected behavior in exception handling is correctly treated and the processor always remains in a known state.

Default memory map

The Armv8-M architecture has a default memory map that is always used even when the MPU Extension is not included. The default memory map distinguishes fixed address regions in memory and allocates default attributes and permissions to these regions. The XN (Execute Never) permission specifies that a region is only allowed to host data. The processor recognized any attempt to fetch instructions from this region as a fault (IACCVIOL MemManage fault).

This Security Mechanism guarantees a minimum and always-on memory protection level.

When the optional MPU Extension is included, the usage of the default memory map is controlled from MPU attributes.

Stack limit checking

The Armv8-M architecture defines a stack limit checking (SPLIM). This guarantees that any attempt to decrease the stack lower than a stack limit value generates a STKOF usage fault. The limit of the main stack is defined in the Main Stack Pointer Limit Register and the limit of the processor stack is defined in the Process Stack Pointer Limit Register. Privileged software can modify these registers.

When the Security Extension is included, additional Stack Limit Registers are included and used for checking the stack limits of Secure and Non-secure software.

This Security Mechanism guarantees protection against stack overflow software attacks.

Undefined instruction

The Armv8-M architecture does not specify the behavior of the processor when UNPREDICTABLE or IMPLEMENTATION DEFINED instructions are executed. However, for the Cortex-M35P processor all these cases are recognized and specifically treated. Details about the specification can be found on the *Arm® Cortex-M35P v8-M Architecture Supplement*.

8.2.2 SM.Bypass-Protect

The Armv8-M architecture adheres to the classic programmers' model with two levels, a privileged level and an unprivileged level. The processor can be in one mode only at a time, but it can switch between modes because of events or programmatically. The model is implemented in a set of registers and instructions. Depending on the current mode, instructions have access to some registers and use a specific register set. The behavior of the model has been designed to resist bypass. For details on the programming model and the rules the implementation adheres to, see the *Arm®v8-M Architecture Reference Manual.* The MPU Extension and the Security Extension use the programming model of the Mainline implementation based on the privileged level and unprivileged level and extend the model with specific sets of registers and behavior in instructions.