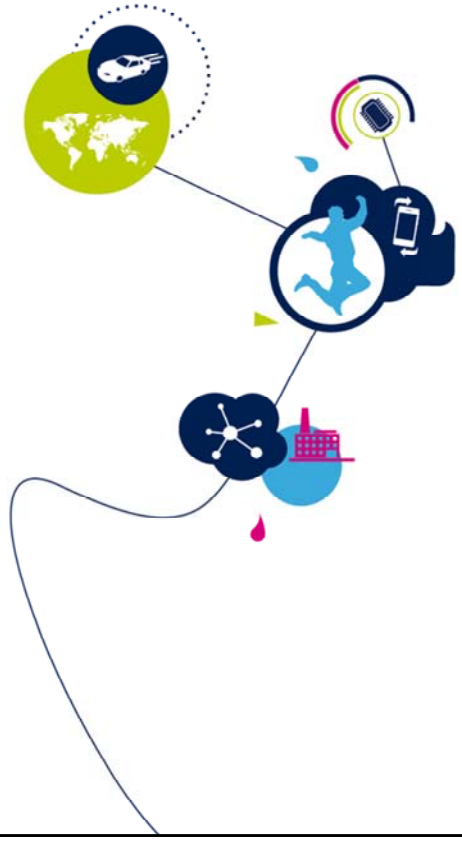
STM32L5 - TRZ

TrustZone® security concept
Revision 1.0



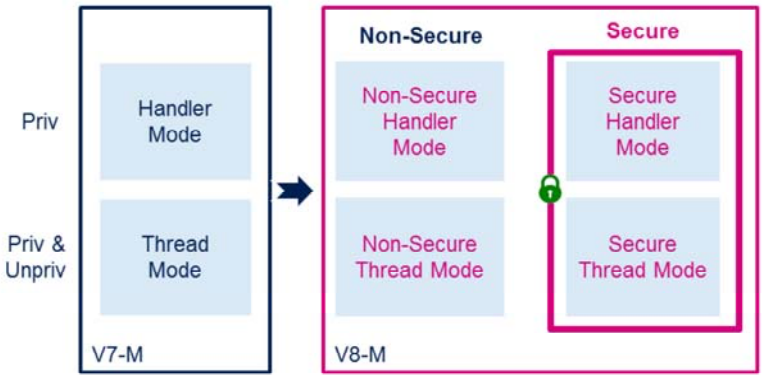
life.augmented

Hello, and welcome to this presentation of the TrustZone security concept, implemented on top of the ARM V8-M architecture compliant Cortex-M33 core.
This presentation applies to an STM32L5 where TrustZone feature has been activated, setting the non-volatile option byte TZEN to 1 (RDP0 only).

Overview

2

- Addition of an extra processor state
  - Secure / non-secure
  - System starts in secure state



Application benefits

- Isolation between non-secure world supporting general purpose microcontroller application and secure world supporting secure applications
- Secure and non-secure partitions running on the same core, thus consuming less than traditional general-purpose MCU and secure element

The ARM V8-M architecture implements TrustZone for Cortex-M cores.
As explained in the figure, the V7-M cores support one kernel running in privileged state, switching unprivileged stacks.
When the security features of the core is enabled, there are 2 orthogonal security states: secure and non-secure.
The non-secure kernel and applications run in non-secure state while secure kernel and applications run in secure state.
Security state of the processor depends on the address at which the instruction was fetched.
Processor always reset to secure state when TrustZone feature is enabled.
Each security state supports both privileged and unprivileged user access.
When TrustZone feature is not enabled the programmers model only include non-secure state.

For each security state the processor can operate in Thread or Handler mode.
At reset or return from an exception, the processor enters in Thread mode – both privileged and unprivileged can run in Thread mode.

TrustZone® Concept

3

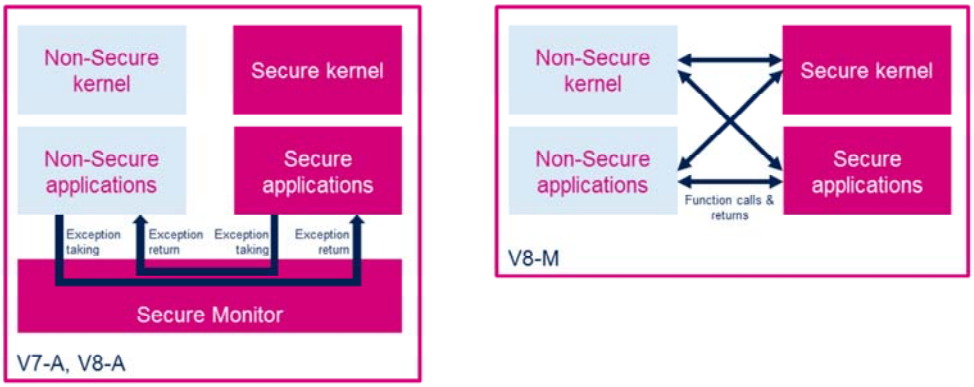
- TrustZone® feature is optional on ARM V8-M core, such as the Cortex® -M33 present in the STM32L5 microcontrollers
  - When enabled, the microcontroller by default starts up in secure state
  - When not enabled, the microcontroller is always in non-secure state
- As for TrustZone® in Cortex® -A processors, code running in Secure state can access both secure and non-secure data
- Unlike TrustZone in Cortex® -A the division of secure and non-secure worlds is memory mapped and the transitions takes place automatically without the need of secure monitor exception handler, thus optimizing switching overhead

life.augmented

TrustZone feature is optional on ARM V8-M core.
When not enabled, the STM32L5 microcontrollers, embedding the Cortex-M33 V8-M compliant core, run in non-secure state, which is identical to ARM V7-M core execution environment.
When enabled, the microcontroller boots in secure state and during runtime can transition from secure to non-secure back and forth.
As for TrustZone in Cortex-A processors, code running in Secure state can access both secure and non-secure data.
Non-secure data accessed by secure software is tagged as non-secure to pass downstream firewalls.
TrustZone for Cortex-A is based on a secure monitor, which is in charge of switching the two secure worlds. So transitions are requested through software exception taking and return.
TrustZone for Cortex-M is based on memory-mapping, enabling direct access to the secure world by using function

calls and returns, thus optimizing switching overhead.

TrustZone®-M versus TrustZone® -A



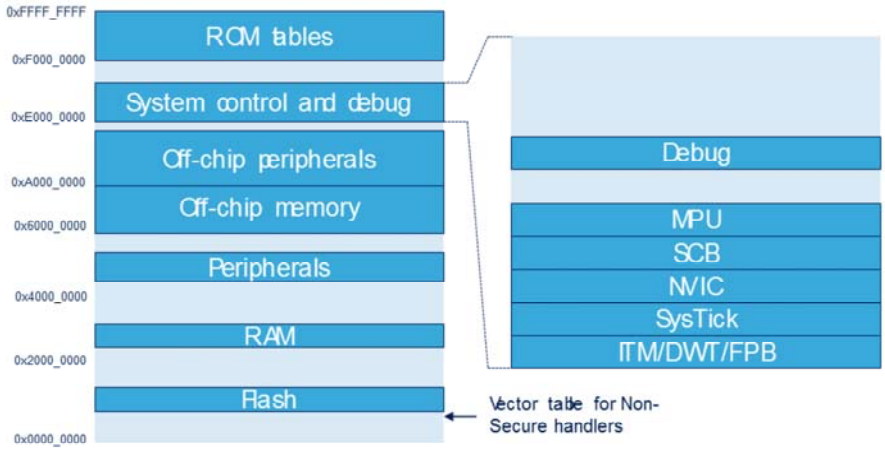
- Conceptually TrustZone for ARM® V8-M is similar to the TrustZone technology found in ARM Cortex®-A Processors.
  - Operations of TrustZone® for ARM® v8-M are however very different as they are optimized for embedded systems that requires real-time responsiveness

life.augmented

TrustZone for ARM V8-M and ARM V8-A have the same objectives: supporting secure and non-secure partitions running on the same core, and ensuring the protection of secure resources.
However the operations of TrustZone for ARMv8-M are very different as they are optimized for embedded systems that requires real-time responsiveness.
Function call and return are used to perform the switch between the two security worlds, instead of exception taking and return.

Non-Secure memory view - TrustZone® Concept

- Non-Secure state memory mapping

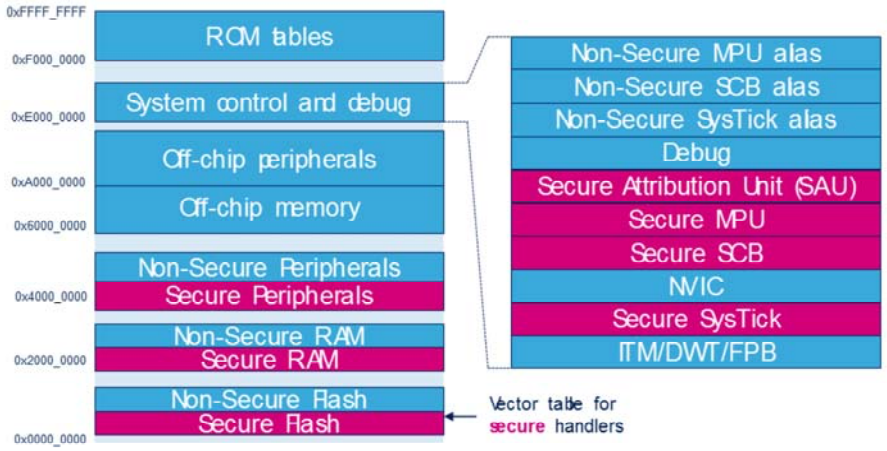


life.augmented

This slide describes the memory mapping used when the core runs in non-secure state.
It is identical to ARM V7-M memory mapping.
Branches to memory locations, defined by secure software and called secure gates, enable non-secure software to call a secure service.
Secure memory is invisible from non-secure software.

Secure memory view - TrustZone® Concept

- Secure state memory mapping



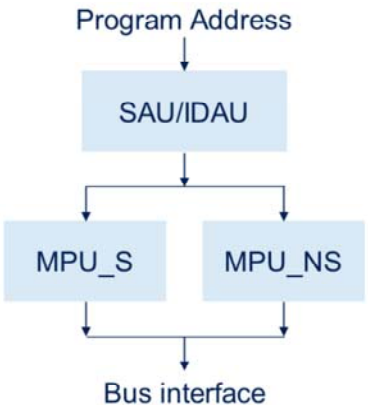
life.augmented

This slide describes the memory mapping used when the core runs in secure state.
Secure memory view shows additional secure flash, RAM and peripherals address ranges.
Access to all regions is possible in secure state.
Since resources such as System Control Block (SCB), Memory Protection Unit (MPU) and SysTick timer are duplicated, one instance being secure, the other one non-secure, the non-secure instances are shifted in the memory mapping by an offset equal to 0x20000.
Secure software can therefore access secure and non-secure instances of these units.
Privileged secure software assigns security attribute to memory regions by programming the Secure Attribution Unit (SAU).

SAU / IDAU - TrustZone® Concept

- Split of secure and non-secure memory map done through
  - SAU (Security Attribution Unit)
  - IDAU (Implementation Defined Attribution Unit)

Unit	Description
IDAU	Hardcoded setting, cannot be changed by software By default active after a reset, while SAU needs to be programmed
SAU	Programmable unit, in charge of assigning security attributes to memory regions Only secure privileged software is permitted to program the SAU 8 regions
MPU	Not compatible with V7-A MPU MPU is banked for Secure and Non-Secure worlds 16 regions (8+8)



life.augmented

Two units determine the security attribute of memory regions:

- The Security Attribution Unit (SAU), which is programmable
- The Implementation Defined Attribution Unit (IDAU), which is hardcoded.

When SAU and IDAU provide different setting for security attribute, the most conservative one is selected.

For example, if SAU maps a region as non-secure and IDAU as secure, the region is tagged as secure.

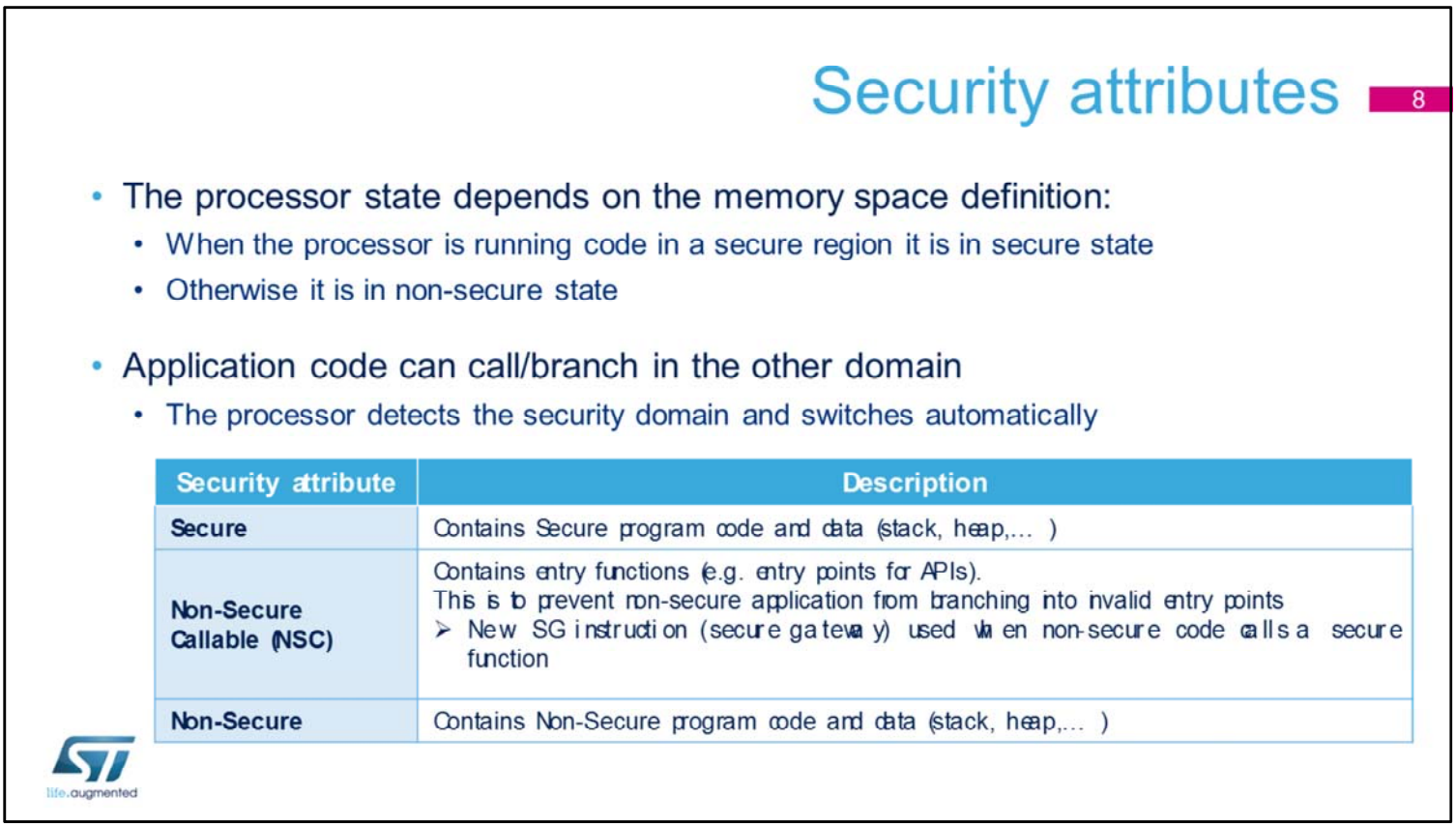
SAU supports 8 regions, defined by start and end addresses.

The alignment requirement for start and end address is 32 bytes.

The V8-M Memory Protection Unit (MPU) is also programmed through start address and end address.

The V7-M MPU is based on start address and power-of-two size. Therefore these two versions of the MPU are not

compatible.
MPU is banked for secure and non-secure worlds.
16 regions per MPU are supported in the STM32L5.



The processor state depends on the memory space definition:
When the processor is running code in a secure region it is in secure state
Otherwise it is in non-secure state.
Inter security domain calls and returns are supported. The processor detects the security domain of the branch address and switches domain automatically.
The ARM V8-M defines the following security attributes:

- Secure region, which contains secure program and data
- Non-Secure Callable region, which is secure but can be called from non-secure state.
- Non-secure region, which contains non-secure program and data.

The Non-Secure Callable region enables non-secure software to call a secure service.
It is declared by secure software and must contain a Secure Gateway (SG) instruction to cause the transition to secure

state.

IDAU

- The IDAU is used to indicate to the processor if a particular memory address is Secure, Secure NSC, or Non-Secure, as well as providing the region number within which the memory address resides
- It can also mark a memory region to be exempted from security checking (e.g. a ROM table)

The IDAU is external to the ARM core.
Through an address decoding logic, it returns to the core the security attribute of the region, as well as its number.
It can also mark a region to be exempted from security checking, such as debug units.
The CoreSight specification, which is a framework for debugging ARM cores, implements another mechanism to assign security attributes to debug units, based on authentication signals, sampled by these units.

Software flow between memory types

- 3 different attribute in SAU/IDAU
  - Secure
  - Non-Secure Callable (NSC)
  - Non-Secure

The secure attribute delivered by the SAU and IDAU can be secure, secure non-secure callable (NSC) and non-secure.
The slide represents the switch from non-secure state to secure state when it is triggered by a non-secure application.
When the non-secure application requests a secure service, it first calls a veneer contained in a secure non-secure callable region.
This veneer must start with a SG instruction.
If these conditions are satisfied, then the transition to secure state is effective.
The veneer generally branches to a secure region, containing the library of secure services.
Once the secure service is completed, a direct transition from secure region back to non-secure region is performed.

Enhanced Memory Protection Unit (MPU)

- PMSAv7 MPU specification requires start address to be multiple of region size this latter one being necessary a power of 2
- New PMSA-v8 allows to define memory regions using start and end address with 32 bytes region granularity

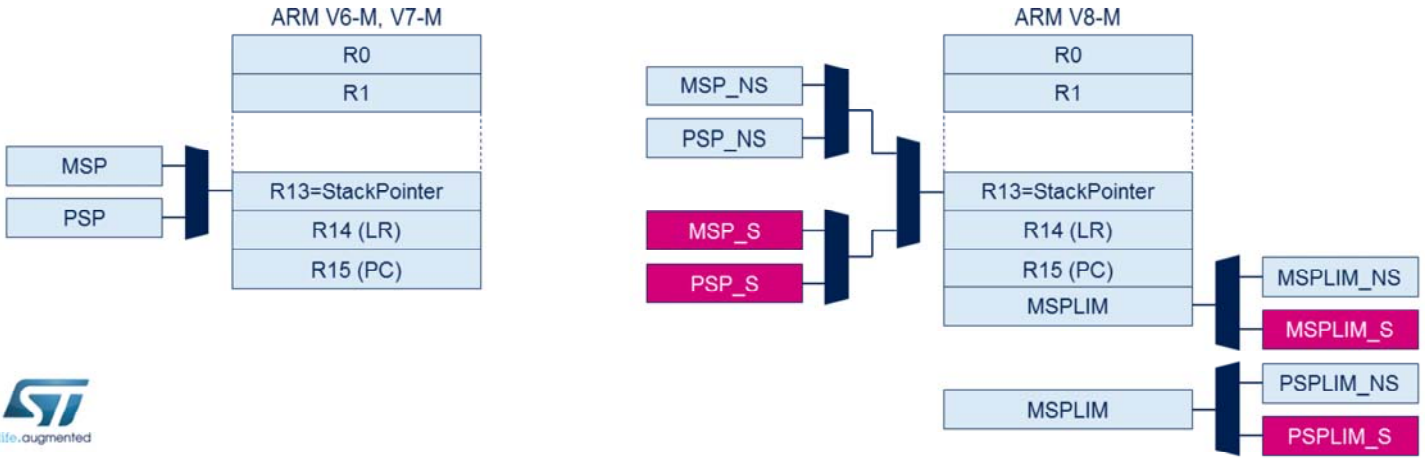
The Protected Memory System Architecture described in ARM V7-a specifies an MPU whose region's size is a power of 2 and region address is a multiple of its size.
The Protected Memory System Architecture described in ARM V8-a specifies a new MPU whose regions boundaries are defined by a start and end address pair. The alignment requirement for start and end address is 32 bytes.
It is much more flexible as the kernel can directly program MPU regions from start and end addresses of the various sections to be mapped.
The STM32L5 supports 8 regions for the secure MPU and 8 regions for the non-secure MPU.
The example, which is provided below, compares the MPU region setting in PMSA v7 MPU and PMSA v8 MPU when the area to be mapped is 274 KB long starting at hexadecimal address 0x0003_BC00.
Due to the power of 2 size and alignment constraint, 4 regions are required for PMSA v7.

One unique region is required for PMSA v8.

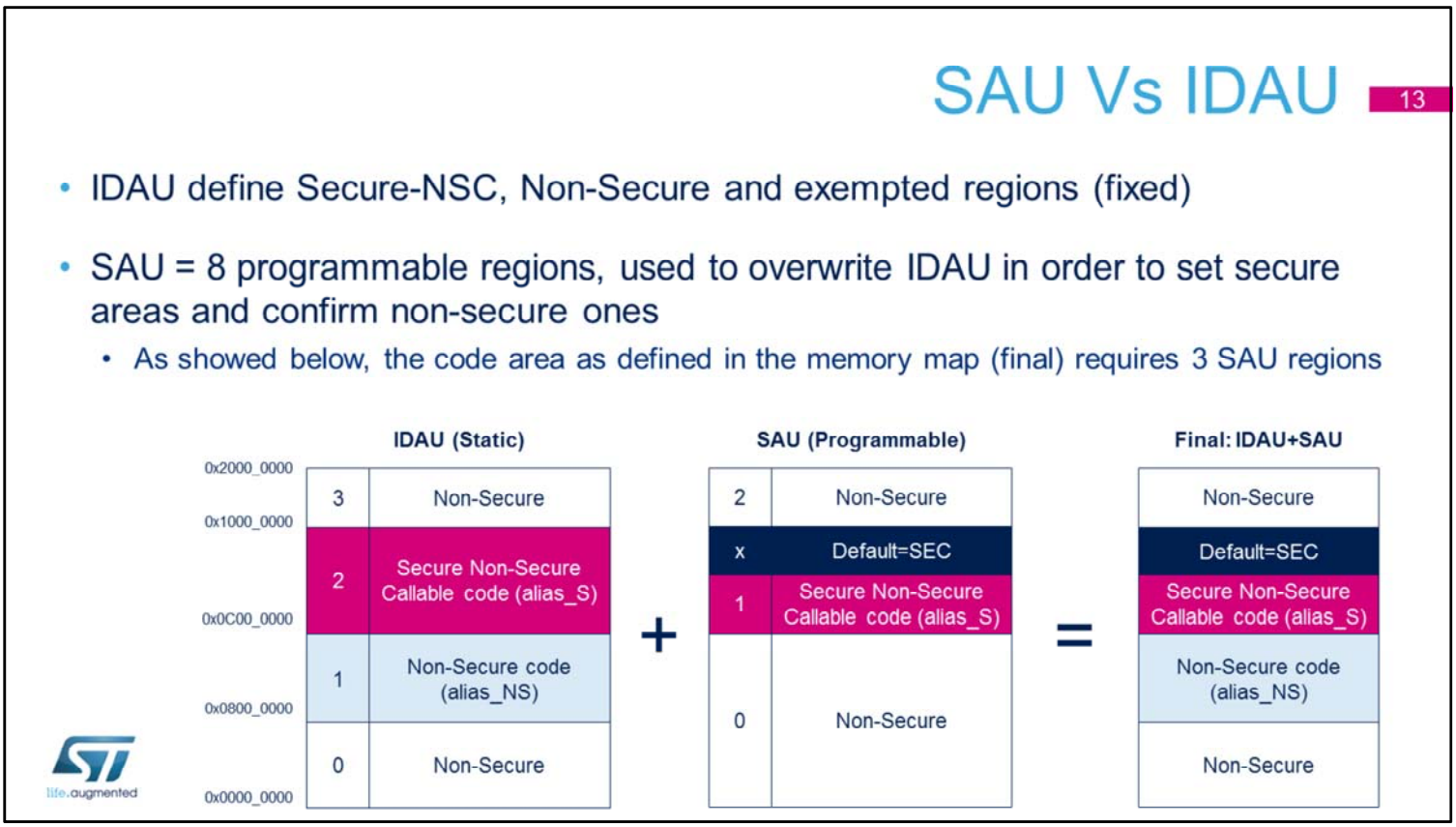
Additional stack pointer resources in V8-M

12

- New pair of Stack Pointers (MSP_S and PSP_S) for the secure stack
- Control register is banked as well as some of the other registers (SCB / MPU / SysTick timer)



The ARM V8-M architecture provides new features regarding stack pointers.
In ARM V7-M, two stack pointers are banked, the Main Stack Pointer (MSP) and Process Stack Pointer (PSP).
The ARM V8-M implements another level of banking, between secure and non-secure states. Globally 4 stack pointers are present, two pairs of MSP and PSP, one per security level.
Furthermore, new stack limit registers enable hardware to detect stack overflow conditions.
Both MSP and PSP stack being protected, two pairs of stack limit registers are implemented, one per security level: non-secure PSP and MSP limit registers and secure PSP and MSP limit registers.
Stack overflow detection generally requires the definition of one non-accessible region per processor stack in V7-M systems.



The STM32L5 security attribution units have the following features:

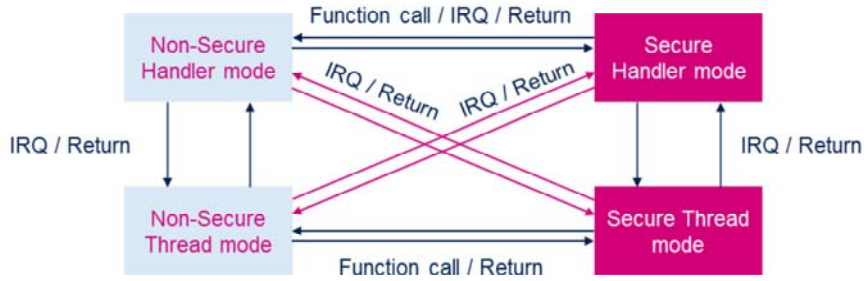
- The fixed IDAU has a granularity of 64 Megabyte and supports the following security attributes: Secure Non-Secure Callable, Non-Secure and exempted regions
- The programmable SAU supports 8 regions featuring the following attributes: Non-Secure or Secure Non-Secure Callable. Unmapped regions are by default secure.

The resulting attribute when IDAU and SAU regions overlap is the most conservative one.

For example in the figure below, when the unmapped SAU region secure overlaps with the Secure Non-Secure Callable IDAU region, the resulting attribute is Secure. So for a region to be non-secure, both IDAU and SAU attributes related to this region have to be non-secure.

Transitions between secure and non-secure worlds in TrustZone V8-M

- If an exception/interrupt has the same state as the current CPU state, then the exception sequence is almost identical to the V7-M processors
- Otherwise (pink arrows), for example when a non-secure interrupt takes place while CPU is running secure code the CPU automatically:
  - Pushes all secure information onto the secure stack
  - Erases the contents from the register banks to avoid leak of secure information



When an interrupt has the same security level as the current processor state, then the exception sequence is almost identical to the V7-M processors. In the slide, it corresponds to the transition from Thread mode to Handler mode, which is handled identically in secure and non-secure states.
When a non-secure interrupt occurs while processor is running in secure state, transition from secure to non-secure state has to be performed, by avoiding any leakage in general purpose registers. This is achieved by automatically pushing all secure information onto the secure stack and erasing the contents from the register banks.
The consequence is an increase of the interrupt latency time, from 12 clocks to 21 clocks.

The slide represents all possible transitions between the four possible processor states. The eight transitions in the middle of the figure cause a change of the security state. Function calls and returns can be used when privilege state remains

the same, for instance from non-secure thread mode to secure thread mode. Otherwise an interrupt entry and return is required.

Vector tables

- In order to handle Secure and Non-Secure exceptions, the core supports two vector tables:
  - VTOR_S (TZEN=1)
  - VTOR_NS

- If security is enabled (TZEN=1), at reset, the CPU sets MSP_S and the Program Counter using the secure vector table
  - Otherwise (TZEN=0), the CPU sets MSP_NS and Program Counter using Non-Secure vector table

life.augmented

When TrustZone is enabled, non-secure exceptions are handled by non-secure kernel and secure exceptions are handled by secure kernel.
Consequently two separate vector tables are implemented.
The two first entries of the vector table are reset vectors, respectively default Main Stack Pointer register value and default Program Counter (PC) register value.
Unlike V7-M cores such as Cortex-M4, the default location of the vector table is not necessarily at address zero.
Since the processor boots in secure state, the register pointing to the secure vector table, called VTOR_S, is initialized from core input signals sampled upon reset.
Then the secure software can program the register pointing to the non-secure vector table, called VTOR_NS, and initialize the two first entries of the non-secure table, MSP_NS and PC_NS in order to emulate a non-secure reset.
When TrustZone is disabled, a unique vector table is used,

because the core only runs in non-secure state.

Vector tables

16

#	Exception	Changes	Security
2	NM	Targets Secure state by default, and can be routed to Non-Secure state by a bit programmable bit	**Configurable**
3	HardFault	Targets Secure state by default, and can be routed to Non-Secure state by a bit programmable bit, except Secure Fault escalation	**Configurable**
4	MemManage	Banked between Secure state and Non-Secure state, depends on the state when the access occurred	Banked
5	BusFault	Targets Secure state by default, and can be routed to Non-Secure state by a programmable bit	**Configurable**
6	UsageFault	Banked between Secure state and Non-Secure state, depends on the state when the access occurred	Banked
7	**SecureFault**	Always targets Secure state	**Secure**
11	SVCall	Banked between Secure state and Non-Secure state, depends on the state when SVC is executed	Banked
12	DebugMonitor	Controllable by hardware configuration (SPIDEN input) to target Secure or Non-Secure state	**Configurable**
14	PendSV	Banked between Secure state and Non-Secure state	Banked
15	SysTick	There are two SysTick timers and the SysTick exception is banked between Secure state and Non-Secure state	Banked
>15	interrupts	Each of the interrupts can be programmed to be Secure or Non-Secure using ne wregister(s) called Interrupt Target Non-Secure state register (NVIC_ITNS) ➢ ITNS can be accessed only in Secure privileged state.	**Configurable**

life.augmented

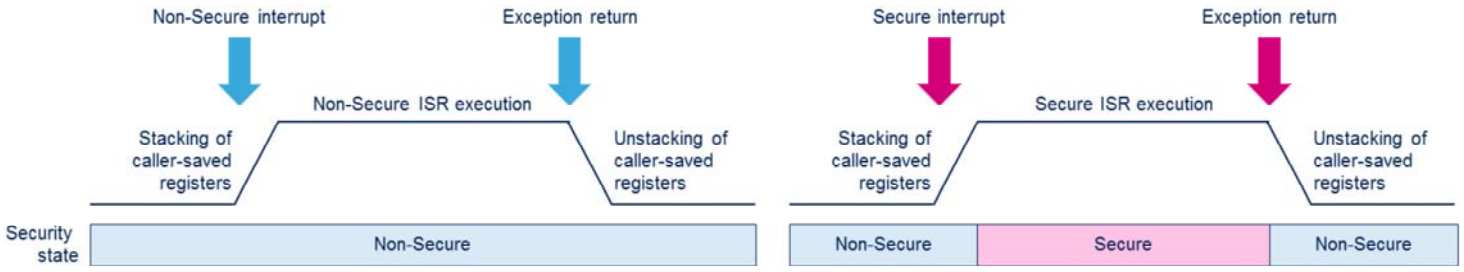
This slide describes the contents of the vector table and indicates whether the corresponding exceptions are secure or non-secure.
Reset vectors, number 0 and 1, explained in the previous slide, are not represented.
Regarding the security state in which the exception will be handled, there are three possibilities:

- Configurable means that secure software can decide what will be the target security state, either non-secure or secure.
- Banked means that the exception is handled in the current security state
- Secure means that the exception is always handled in secure state.

For example, the Non-Maskable Interrupt (NMI) is either handled in secure state or in non-secure state, according to a programmable control bit present in a secure only register.
A banked exception is taken in the secure state in which it

occurs. For example the memory management fault, which is taken in case of access permission violation, is handled independently in each security domain.
A new system exception called SecureFault is introduced in the ARMv8-M Mainline for handling of security violations. It always targets the secure state.

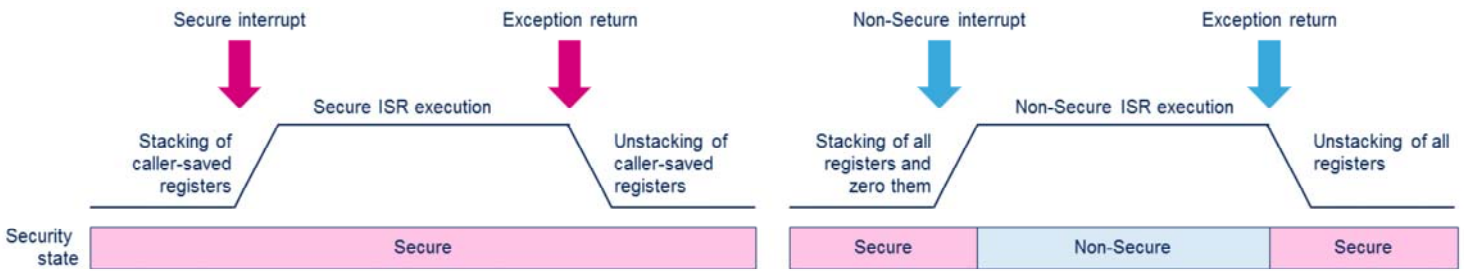
Exceptions taken while current state is non-secure



- Caller-saved registers are stacked automatically as in previous Cortex-M processors

When an interrupt is taken while the current security state is non-secure and the interrupt source is non-secure, the sequence is the same as V7-M exception mechanism. The caller-saved registers are automatically saved when exception is taken and automatically restored upon execution of the exception return instruction. The Interrupt Service Routine is executed in non-secure state.
When an interrupt is taken while the current security state is non-secure and the interrupt source is secure, the unique difference is that the Interrupt Service Routine is executed in secure state. Caller-saved registers are pushed and popped to / from the non-secure stack.

Exceptions taken while current state is secure



- When transitioning from Secure to Non-Secure state, to avoid any leak of the secure information, the CPU will stack all registers and clear them (21 cycles)

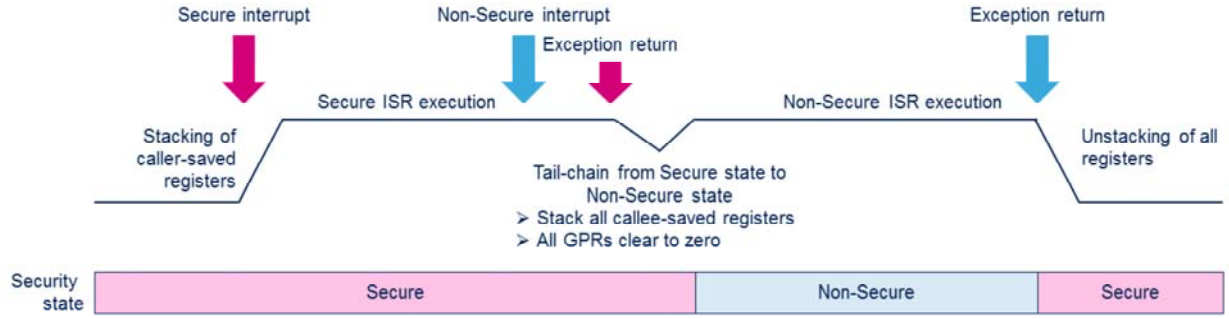
When an interrupt is taken while the current security state is secure and the interrupt source is secure, the sequence is the same as V7-M exception mechanism. The caller-saved registers are automatically saved when exception is taken and automatically restored upon execution of the exception return instruction. The Interrupt Service Routine (ISR) is executed in secure state.

When an interrupt is taken while the current security state is secure and the interrupt source is non-secure, then a risk of register contents leak exists, because the general-purpose registers contain secure data.

Therefore the microcode in charge of register stacking saves all registers into the secure stack and then writes zero to all of them, prior to starting the execution of the non-secure ISR. 21 clock cycles are needed to achieve this register stacking.

Unstacking is performed from the secure stack and restores the contents of all general purpose registers.

Tail chaining operation



- When secure ISR is interrupted by non-secure ISR, the CPU will also consider security by clearing all registers before executing the non-secure interrupt handler

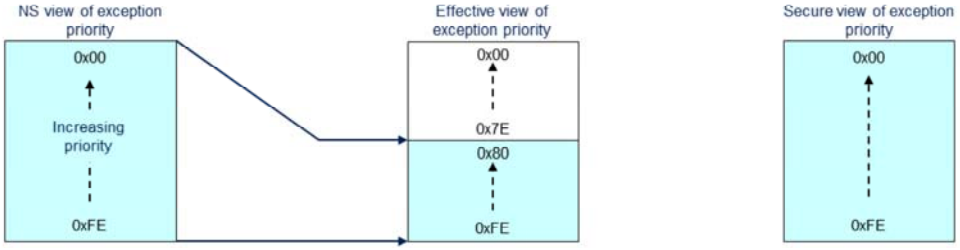
life.augmented

This sequence details the tail-chaining from a secure interrupt service routine to a non-secure interrupt service routine.
At the beginning, the processor runs in secure state, while a secure interrupt is requested.
The microcode saves to the secure stack the caller-saved registers.
Then a non-secure interrupt is requested.
When the exception return instruction is executed at the end of the secure ISR, the microcode in charge of tail-chaining automatically saves to the secure stack all callee-saved registers, so that all general purpose registers (GPR) have been saved.
All GPRs are automatically cleared to zero prior to starting the execution of the non-secure interrupt service routine.
At the end, when the exception return instruction is executed, the microcode restores all GPRs from the secure stack before resuming the execution of the secure

application.

Exception/Interrupt Priority

- Secure and Non-Secure interrupt priorities can share the same levels
  - Optionally, a new programmable bit in AIRCR called PRIS can be set to place all Non-Secure configurable exceptions in lower priority level space so that Secure interrupts can potentially have higher priorities



- The following exception/interrupt masking registers are also banked
  - PRIMASK, FAULTMASK, BASEPRI

life.augmented

Both secure and non-secure software are in charge of assigning a priority to any exception source.
The lower the value, the higher the priority.
A security issue could happen if the non-secure software selects high priority levels (priority value close to 0) in order to mask the secure interrupts.
The V8-M features a secure control bit, called PRIS, that shifts the non-secure programmed priority level one bit to the right, inserting a one onto the left. The consequence is that effective non-secure priority levels are in range 0x80 to 0xFE. Thus any secure interrupt can be programmed with a higher priority than non-secure ones.
The V7-M priority boosting registers PRIMASK, FAULTMASK and BASEPRI, used to set the execution priority level, are banked in V8-M mainline between secure and non-secure states.

TrustZone®-aware and secureable blocks

Secure block	TZSC, OTFDEC
TrustZone® aware master blocks	Cortex® M33, DMA1, DMA2
TrustZone® aware slave blocks	GTZC, EXTI, RCC, DMAMUX, APB1 bridge, APB2 bridge, GPIOA-H, PWR, SYSCFG, RTC, TAMP, embedded Flash
Secureable peripherals	OCTOSPI1 registers, FMC registers, SDMMC1, PKA, RNG, HASH, AES, ADC1-2, ICACHE registers, TSC, CRC, DFSDM1, SAI1-2, TIM1-8,15-17, COMP,VREFBUFUCPD1, USB FS, FDCAN1, LPTIM2-3, I2C1-4, LPUART1, LPTIM1, OPAMP, DAC1-2, CRS, UART4-5, USART1-3, SPI1-3, IWDG, WWDG

When the TrustZone security is active, a peripheral can be either Securable or TrustZone-aware type as follows:

• Securable: a peripheral is protected by an AHB/APB firewall gate that is controlled from TZSC controller to define security properties.

• TrustZone-aware: a peripheral connected directly to AHB or APB bus and is implementing a specific TrustZone behavior such as a subset of registers being secure.

TrustZone-aware AHB masters always drive HNONSEC signal according to their security mode (as CM33 core and DMA).

The TrustZone® security controller (TZSC), which is a part of the module called GTZC, defines the secure/privilege state of slave/master peripherals. Therefore, the TZSC itself is a secure block.

Writes to On-the-fly decryption engine (OTFDEC) are always secure when TrustZone is enabled.

Example of memory map

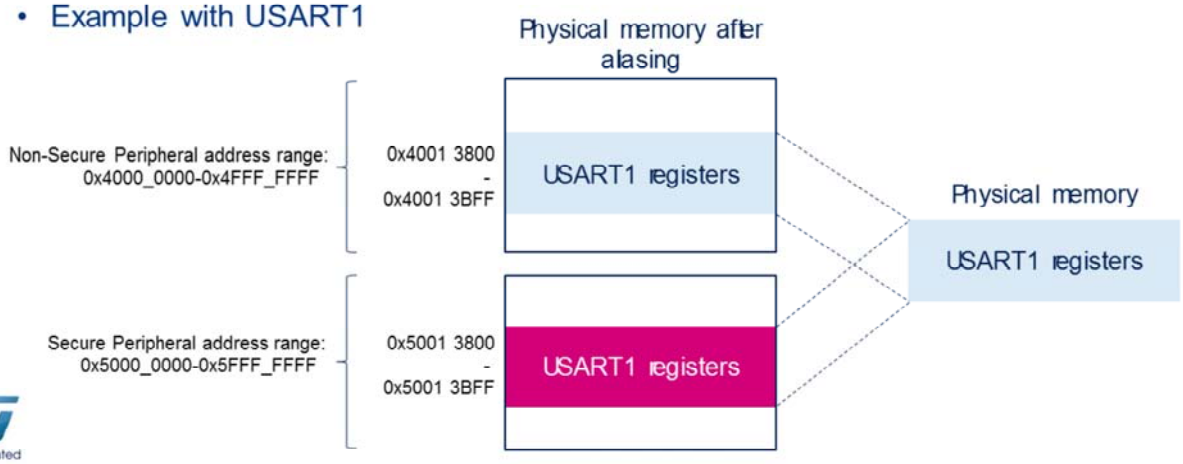
Region description	Address range	IDAU security attribution	SAU security attribution typical configuration	Final security attribution
Code- External memories	0x0000_0000 0x07FF_FFFF	Non-Secure	Secure or Non-Secure or NSC	Secure or Non-Secure or NSC
Code- Flash and SRAM	0x0800_0000 0x08FF_FFFF	Non-Secure	Non-Secure	Non-Secure
	0x0C00_0000 0x0FFF_FFFF	NSC	Secure or NSC	Secure or NSC
Code- External memories	0x1000_0000 0x17FF_FFFF	Non-Secure	Non-Secure	Non-Secure
	0x1800_0000 0x1FFF_FFFF	Non-Secure		
SRAM	0x2000_0000 0x2FFF_FFFF	Non-Secure		
	0x3000_0000 0x3FFF_FFFF	NSC	Secure or NSC	Secure or NSC
Peripherals	0x4000_0000 0x4FFF_FFFF	Non-Secure	Non-Secure	Non-Secure
	0x5000_0000 0x5FFF_FFFF	NSC	Secure or NSC	Secure or NSC
External memories	0x6000_0000 0xDFFF_FFFF	Non-Secure	Secure or Non-Secure or NSC	Secure or Non-Secure or NSC

life.augmented

This table shows an example of typical eight SAU regions mapping on top of IDAU fixed regions.
Based on IDAU security attribution, the Flash, system SRAMs and peripherals memory space is aliased twice for secure and non-secure state.
However, the external memories space is not aliased.
The user can split and choose the secure, non-secure or NSC regions for external memories as needed.

Address aliasing

- The number of SAU regions (8) does not enable the user to select for each peripheral its security state
  - Address aliasing is implemented to map the peripheral at two different addresses, one in a secure region, the other in a non-secure region
  - Example with USART1



In the STM32L5, the security state of internal memories and internal peripherals is not hardcoded. It is programmable in the GTZC module.
Consequently the address range containing peripheral registers or portion of internal memories can be programmed as being secure or non-secure. This is achieved in the SAU. However the number of regions supported by the SAU, which is only 8, does not enable the user to select the security attribute for each peripheral and regions of internal memories.
Address aliasing is implemented to map the peripheral or memory address ranges at two different addresses, one in a secure region, the other in a non-secure region.
The slide describes how the USART1 address range is aliased. In the physical memory space, the USART1 has a unique address range. However it is visible from internal masters at two different addresses: 0x4001_3800 when USART1 is programmed as a non-secure peripheral,

0x5001_3800 when it is programmed as a secure peripheral. Thus the original address in the physical memory is aliased to two different addresses.

Instruction fetches

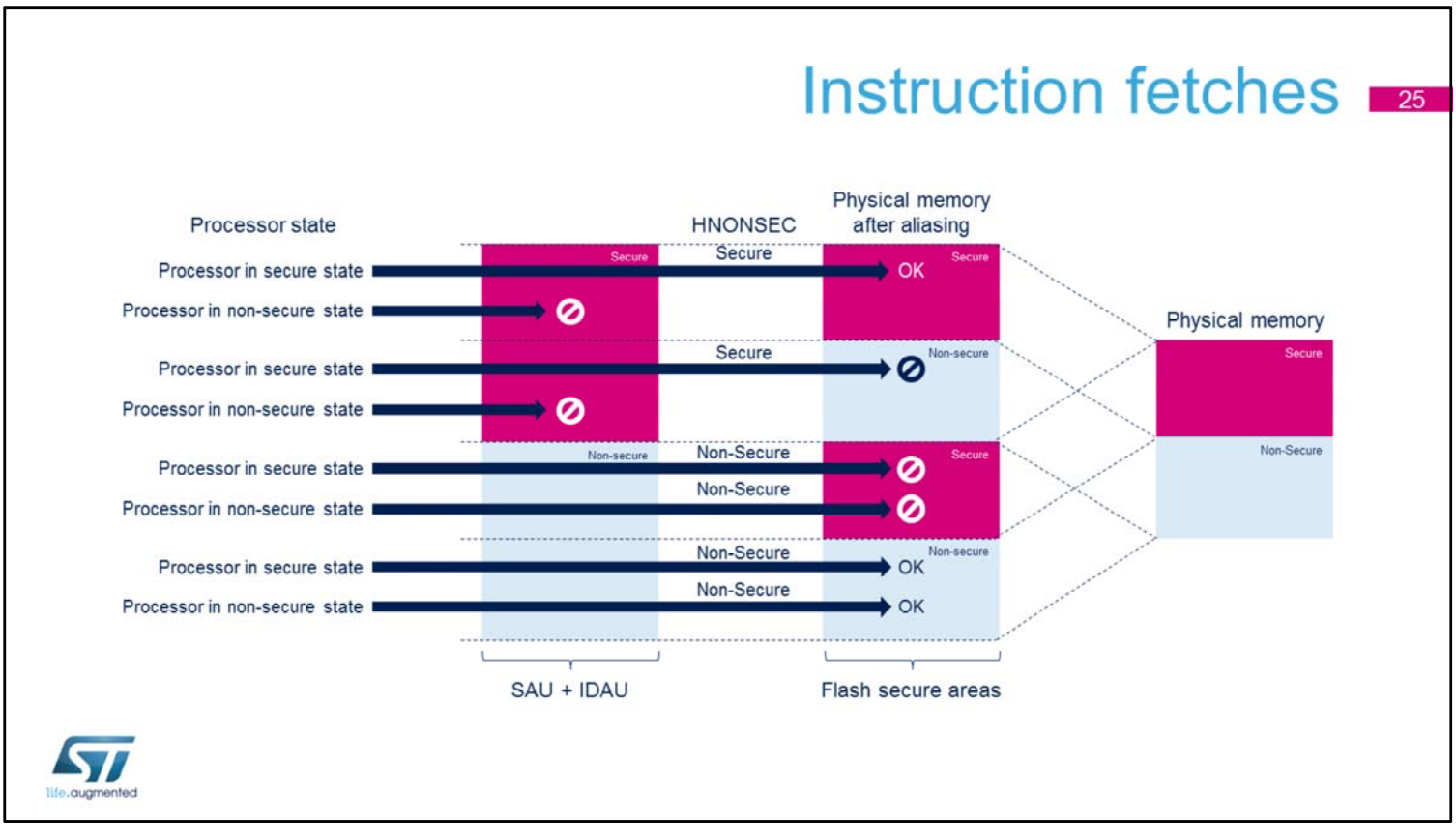
- The AHB5 bus protocol indicates the security attribute of a transaction through the HNONSEC signal
  - The HNONSEC behaviour for instruction fetches is dependent on target address

CPU current security state	Targeted memory (SAU/IDAU)	HNONSEC for instruction fetches	Note
X (don't care)	NS	1 (NS)	Subject to MPU_NS permission
X (don't care)	SEC-NSC	0 (SEC)	Subject to MPU_S permission
X (don't care)	SEC	0 (SEC)	Subject to MPU_S permission

  - Instruction fetches to Secure / Secure NSC is allowed even if CPU runs in non-secure state because it might be entering secure IRQ

life.augmented

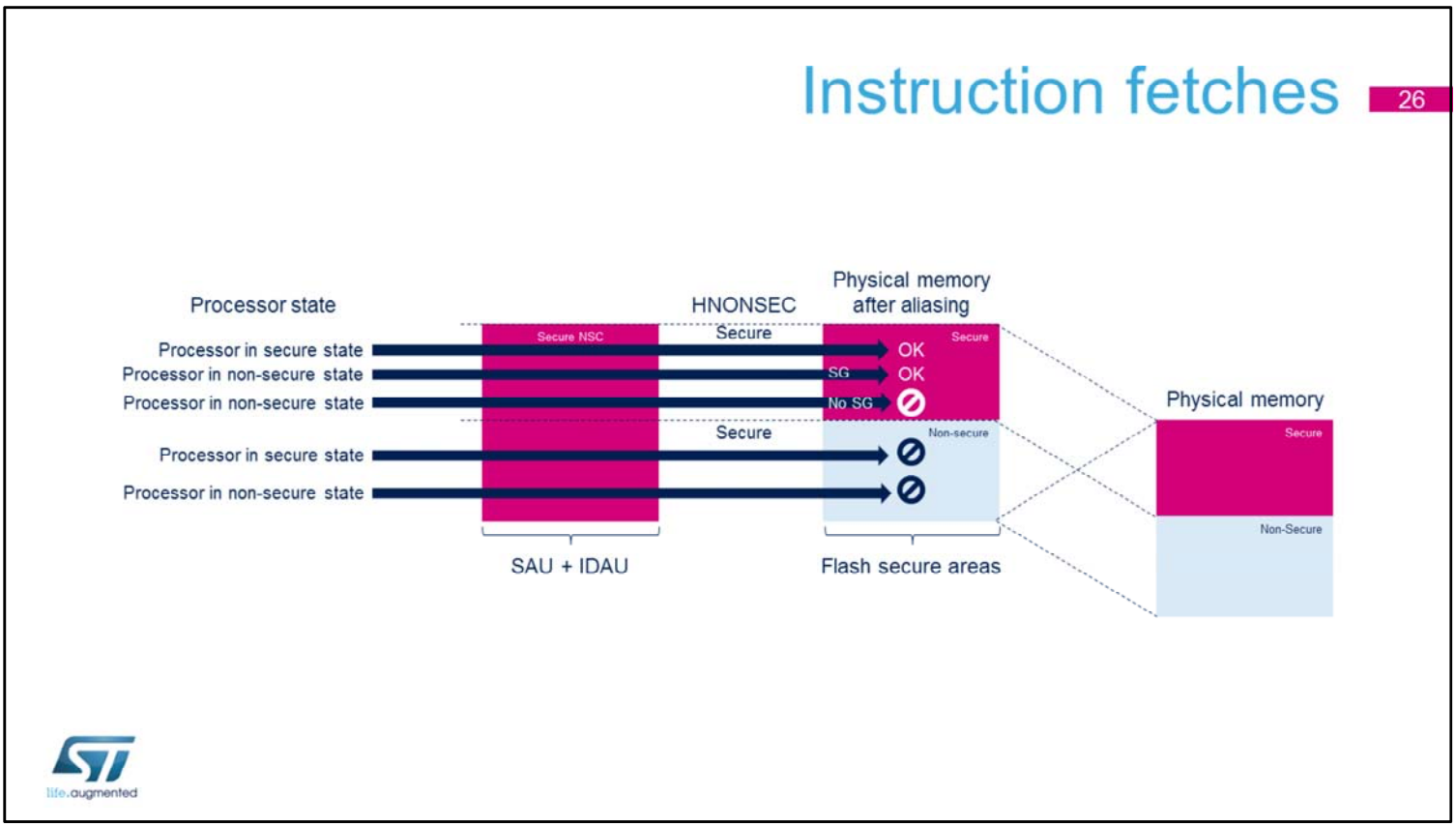
The AHB5 protocol indicates the security attribute of a transaction request through the HNONSEC signal.
This table provides the security attribute of the transaction output on AHB5 bus when the processor fetches instructions, according to the security attribute of the targeted memory defined by the combination of SAU and IDAU settings.
The transaction generated over the AHB5 bus is marked as secure when the processor performs an instruction fetch into a region defined as secure or secure non-securable callable by the combination of SAU and IDAU attributes.
Note that the CPU current security state is not taken into account to define the instruction fetch security attribute.



This figure summarizes the various protections that check the secure attribute of instruction fetches.
Two levels of protection are implemented, one is the SAU/IDAU security attribution units, the other one is the definition of secure areas in the flash memory.
The following instruction fetches are successful:

- Running in secure state and fetching from a region marked as secure in both SAU/IDAU and flash controller
- Running in non-secure state and fetching from a region marked as non-secure in both SAU/IDAU and flash controller.

All other instruction accesses are blocked, either by SAU/IDAU or by the flash security protection.
In particular, a processor running in secure state is not permitted to fetch instruction from a non-secure area.



This figure is slightly different than the previous one, in that the SAU/IDAU declare the region as secure non-secure callable.
In this case, any instruction access initiated in secure or non-secure state is accepted by the SAU/IDAU protection and the transaction is marked as secure on AHB5 bus.
When the target area is programmed as non-secure in the flash controller, the transaction is blocked and an error is signaled.
When the target area is programmed as secure in the flash controller, the transaction is accepted and transition from non-secure to secure is effective only when the instruction called by the non-secure software is a Secure Gate (SG). If the first instruction accessed in the secure non-secure callable area is not SG, the transition to secure state is not performed.

Data access

- Data access excluding stacking, unstacking, vector fetches, debug accesses

CPU current security state	Targeted memory (SAU/IDAU)	HNONSEC for data R/W	Note
NS	NS	1 (NS)	Subject to MPU_NS permission
NS	SEC / SEC-NSC	-	Blocked by SAU/IDAU
SEC	NS	1 (NS)	Subject to MPU_NS permission
SEC	SEC / SEC-NSC	0 (SEC)	Subject to MPU_S permission

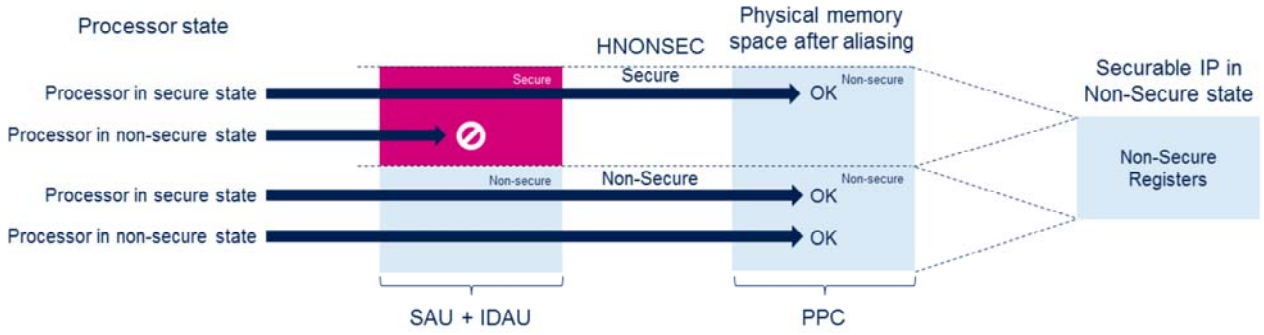
life.augmented

This table summarizes the rules which define the AHB5 transaction type according to target address space when the processor accesses data.
When the processor runs in non-secure state, the data transactions are marked as non-secure.
When it runs in secure state, the data transactions are marked as either secure or non-secure, depending on the SAU/IDAU setting.
Note that enabling secure software to access non-secure address spaces is mandatory to share memory. One important use case is encrypting in a secure state a non ciphered buffer allocated by non-secure software.
When the processor runs in secure state, and SAU/IDAU map the region as non-secure, the data transaction on AHB5 is tagged as non-secure, because the data accessed by the secure software are non-trusted and secure firewalls shall not be passed.
Non-secure access towards secure area is automatically

blocked at SAU/IDAU level.

Data access

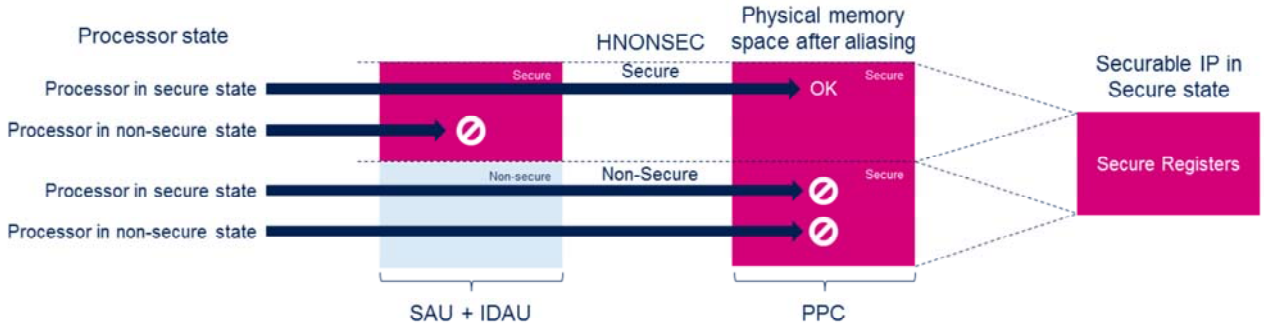
Peripheral Protection Controller (PPC) with Secureable IP



This slide details the security protection mechanisms implemented when the processor accesses a securable IP programmed as non-secure in the Peripheral Protection Controller of the GTZC.
If non-secure software uses the secure aliased address, the SAU/IDAU units block the transaction.
If non-secure software uses the non-secure aliased address, the transaction is successful.
Note that secure software can use two different addresses to access the non-secure peripheral: the one in the secure aliased address range and the one in the non-secure aliased address range.
When secure software uses the non-secure alias, the transaction that occurs on AHB5 is tagged as non-secure.

Data access

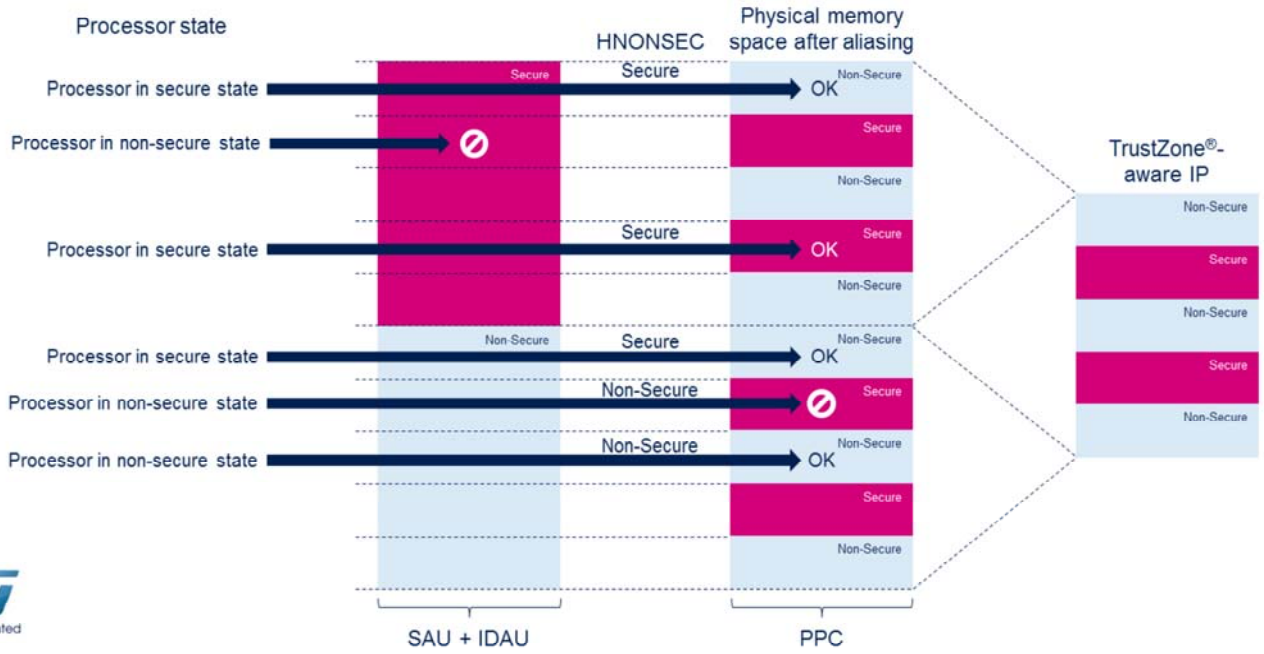
Peripheral Protection Controller (PPC) with Secureable IP



This slide details the security protection mechanisms implemented when the processor accesses a securable IP programmed as secure in the Peripheral Protection Controller of the GTZC.
Any attempt to access this peripheral in non-secure state results in a failure.
The transaction is successful only when the processor runs in secure state and uses the secure alias region.

Data access

Peripheral Protection Controller (PPC) with TrustZone® aware IP



This slide details the security protection mechanisms implemented when the processor accesses a TrustZone-aware IP, containing both secure and non-secure address ranges.

Only secure software is permitted to access the secure parts of the peripheral by using the secure alias.

Both secure software and non-secure software are permitted to access the peripheral registers through the non-secure alias.

However the security protection implemented in the TrustZone-aware peripherals prevents non-secure software from accessing secure address ranges.

Related peripherals

31

- Refer to these peripheral trainings linked to this module
  - Global TrustZone® Controller (GTZC)
  - Embedded Flash memory (Flash)

The Flash memory module has relationships with the following other module:

- Global TrustZone® Controller (GTZC)
- Embedded Flash memory (Flash).