AN5347

Application note

Introduction to Arm® TrustZone® features on STM32L5, STM32U5, and STM32U3 MCUs

Introduction

In IoT (internet of things) applications, devices are vulnerable to unwanted intrusions through the Internet. Consequently, security is an important topic to protect device and information and to isolate the trusted and untrusted worlds from each other.

STM32L5, STM32U5, and STM32U3 series devices are based on the high-performance Arm® Cortex®-M33 32-bit RISC core. This processor uses the Armv8-M architecture and is primarily for environments where security is an important consideration.

The Arm® TrustZone® technology for Armv8-M is a security extension that is designed to partition the hardware into secure and non-secure worlds. With the Arm® TrustZone® technology and software method, the STM32L5/U5/U3 microcontrollers (MCUs) provide a secure application with good design flexibility.

This document introduces the Arm® TrustZone® technology and the features of STM32L5/U5/U3 devices that allow the partition of MCU memory/resources between secure and nonsecure.

1 General information

This application note applies to the STM32L5, STM32U5, and STM32U3 Series microcontrollers that are Arm®Cortex® core-based devices.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Reference documents

[1] Reference manual *STM32L552xx and STM32L562xx advanced Arm®-based 32-bit MCUs* (RM0438)

[2] Reference manual *STM32U575xx and STM32U585xx advanced Arm®-based 32-bit MCUs* (RM0456)

[3] Reference manual *STM32U3 series Arm®-based 32-bit MCUs* (RM0487)

[4] Armv8-M Architecture Reference Manual available from the Arm® web site.

[5] User manual *Evaluation board with STM32L552ZE MCU* (UM2597)

[6] User manual *Evaluation board with STM32U575AII6Q MCU* (UM2854)

[7] User manual *Discovery kit with STM32L562QE MCU* (UM2617)

[8] User manual *Discovery kit for IoT node with STM32U585AI* (UM2839)

[9] User manual *STM32L5 Nucleo-144 board* (UM2581)

[10] User manual *STM32U5 Nucleo-144 board* (UM2861)

2 Arm TrustZone technology

2.1 Overview

The Arm TrustZone technology for Armv8-M partitions the system into two regions: one is secure world and another is nonsecure world.

The division of secure and nonsecure worlds is memory-map based.

All the available microcontroller resources including flash memory, SRAM, external memories, peripherals and interrupts, are allocated to either the secure or nonsecure world. After planning the security attribution of these resources, nonsecure world only accesses nonsecure memories and resources, while secure world is able to access all memories and resources in both worlds, including secure and nonsecure resources.

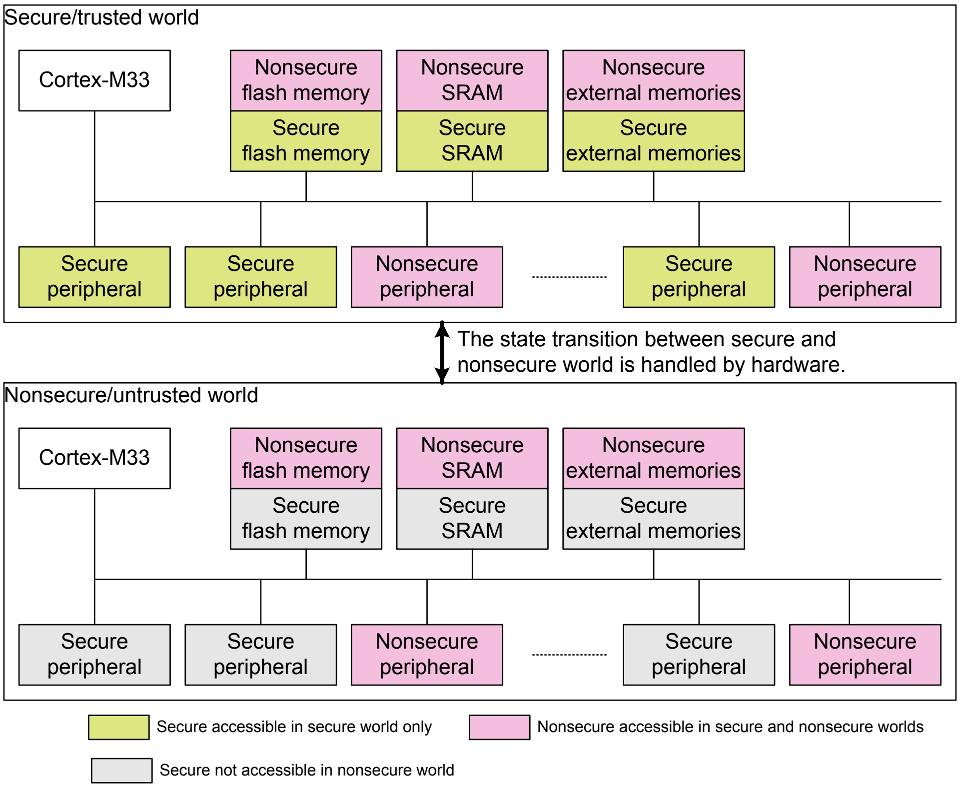
Important data that needs protection (such as cryptographic keys) must be placed and processed safely in the secure world .

The location where the code is executed defines its type:

- When the code is executed in secure memory, it is called secure code.
- When the code is executed in nonsecure memory, it is called nonsecure code.

Secure and nonsecure codes run on the same STM32L5/U5/U3 device, as illustrated in the figure below.

Figure 1. Resource partition between secure and nonsecure worlds



2.2 Security states

In a simplified view, the executed code address determines the security state of the CPU, that is either secure or nonsecure:

- If the CPU runs code in a nonsecure memory, the CPU is in nonsecure state.
- If the CPU runs code in a secure memory, the CPU is in secure state.

The Armv8-M technology defines the following address security attributes:

- **Secure**
  Secure addresses are used for memory and peripherals that are only accessible by secure code or secure masters. Secure transactions are those that originate from masters operating as secure.
- **Nonsecure callable (NSC)**
  NSC is a special type of secure location. This type of memory is the only type for which an Armv8-M processor permits to hold an SG (secure gateway) instruction that enables software to transition from non-secure to secure state. This SG instruction can be used to prevent nonsecure applications from branching into invalid entry points.
  When a nonsecure code calls a function in the secure side:
  – The first instruction in the API must be an SG instruction.
  – The SG instruction must be in an NSC region.

  Secure code also provides nonsecure callable functions to provide secure service accesses to non-secure code.
- **Nonsecure**
  Nonsecure addresses are used for memories and peripherals accessible by all software running on the device. Nonsecure transactions originate from masters operating as nonsecure or from secure masters accessing a nonsecure address (data transaction only, not fetch instructions). Nonsecure transactions are only permitted to access nonsecure addresses. Nonsecure transactions cannot access to secure addresses.

3 TrustZone implementation on STM32L5, STM32U5, and STM32U3 series

3.1 Activation of STM32L5/U5/U3 TrustZone

In STM32L5/U5/U3 devices, the TrustZone is disabled by default and enabled by setting the TZEN option bit in the corresponding option byte.

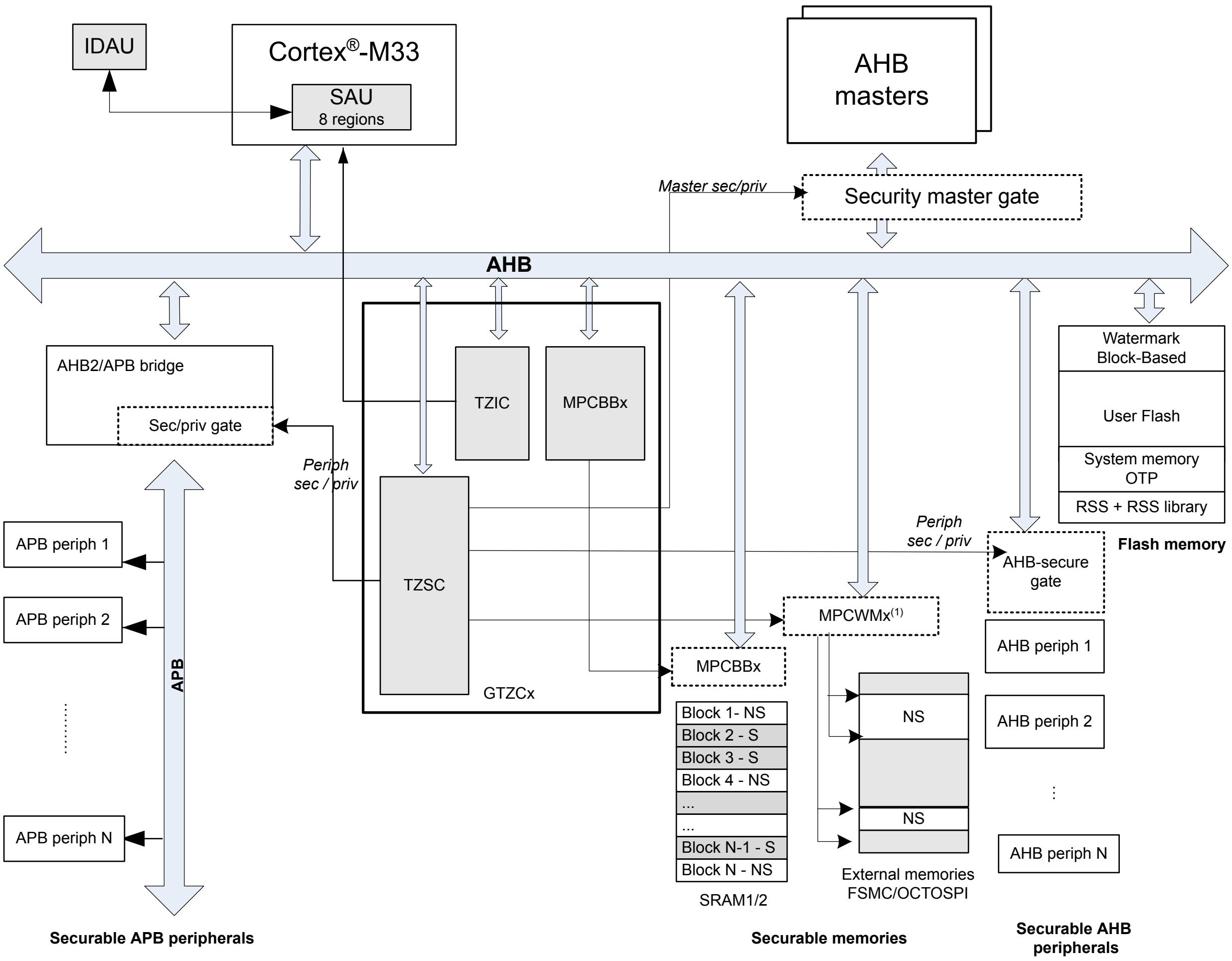
All features described in this document apply to the STM32L5/U5/U3 devices with TrustZone enabled.

3.2 TrustZone block diagram

In STM32L5/U5/U3 devices, the TrustZone is implemented thanks to the SAU (secure attribution unit), the IDAU (implementation defined attribution unit), the flash memory, and the GTZC (global TrustZone security controller).

The block diagram below details the TrustZone implementation.

Figure 2. STM32L5/U5/U3 TrustZone implementation overview



(1) MPCWMx is not available on STM32U3 devices.

3.3 Secure attribution unit (SAU) and implementation defined attribution unit (IDAU)

The security state of a memory address, seen by the CPU, is controlled by a combination of the internal SAU (secure attribution unit) and the IDAU (implementation defined attribution unit).

The security attribution result is the higher security setting between IDAU and SAU. The priority of security attribution is as follows:

- Secure has the highest secure priority.
- Nonsecure callable has lower secure priority.
- Nonsecure has the lowest secure priority.

The table below shows how to assign a specific security attribute (secure, nonsecure or nonsecure callable) to a specific address.

Table 1. Configuring security attributes with IDAU and SAU

IDAU security attribution	SAU security attribution[1]	Final security attribution
Nonsecure	Secure	Secure
	Secure - NSC	Secure - NSC
	Nonsecure	Nonsecure
Secure or NSC[2]	Secure	Secure
	Nonsecure	Secure - NSC

1. *Defined regions are aligned to 32-byte boundaries.*
2. *NSC = nonsecure callable.*

3.3.1 STM32L5/U5/U3 IDAU and memory aliasing

The STM32L5/U5/U3 memory mapping follows the Arm recommendations to implement a duplicated memory map, one for the secure view and the other for the nonsecure view.

This means that each region of the memory map (code, SRAM, peripherals) is divided into two subregions where internal memories and peripherals are decoded at two separate address locations, in the nonsecure view and in the secure view. An IDAU is implemented to define the security attributes of these regions.

The IDAU memory-map partition is not configurable. It is fixed by hardware. The table below shows the memory-map security-attribution partition defined by the STM32L5/U5/U3 IDAU.

Table 2. IDAU memory map address security attribution on STM32L5, STM32U5, and STM32U3

Region	Address range	Security attribute through IDAU
Code-external memories when remapped	0x0000 0000-0x07FF FFFF (128 Mbytes)	Nonsecure
Code-flash memory and SRAM	0x0800 0000-0x0BFF FFFF (64 Mbytes)	Nonsecure
	0x0C00 0000-0x0FFF FFFF (256 Mbytes)	Nonsecure callable
Code-external memories when remapped	0x1000 0000-0x1FFF FFFF (256 Mbytes)	Nonsecure
SRAM	0x2000 0000-0x2FFF FFFF (256 Mbytes)	Nonsecure
	0x3000 0000-0x3FFF FFFF (256 Mbytes)	Nonsecure callable
Peripherals	0x4000 0000-0x4FFF FFFF (256 Mbytes)	Nonsecure
	0x5000 0000-0x5FFF FFFF (256 Mbytes)	Nonsecure callable
External memory[1]	0x6000 0000-0xDFFF FFFF (2 Gbytes)	Nonsecure

1. *The external memory area is not aliased.*

3.3.2 STM32L5/U5/U3 SAU

There are eight SAU regions in the STM32L5/U5/U3. The user changes the required security configuration partition by SAU as shown in the table below. When the TrustZone is enabled, the SAU defaults all addresses as secure: all memory regions are considered as secure.

Table 3. SAU memory map address security attribution on STM32L5, STM32U5, and STM32U3

Region	Address range	Security attribute through IDAU	Security attribute through SAU	Final security attribute
Code-external memories when remapped	0x0000 0000 - 0x07FF FFFF	Nonsecure	Secure Nonsecure or Nonsecure callable	Secure Nonsecure or Nonsecure callable
Code-flash memory and SRAM	0x0800 0000 - 0x0BFF FFFF	Nonsecure	Nonsecure	Nonsecure
	0x0C00 0000 - 0x0FFF FFFF	Nonsecure callable	Secure or Nonsecure callable	Secure or Nonsecure callable
Code-external memories when remapped	0x1000 0000 - 0x1FFF FFFF	Nonsecure	Nonsecure	Nonsecure
SRAM	0x2000 0000 - 0x2FFF FFFF	Nonsecure	Nonsecure	Nonsecure
	0x3000 0000 - 0x3FFF FFFF	Nonsecure callable	Secure or Nonsecure callable	Secure or Nonsecure callable
Peripherals	0x4000 0000 - 0x4FFF FFFF	Nonsecure	Nonsecure	Nonsecure
	0x5000 0000 - 0x5FFF FFFF	Nonsecure callable	Secure or Nonsecure callable	Secure or Nonsecure callable
External memories	0x6000 0000 - 0xDFFF FFFF	Nonsecure	Secure Nonsecure or Nonsecure callable	Secure Nonsecure or Nonsecure callable

Example

A peripheral is decoded at two address ranges: 0x4000 0000 in nonsecure view and 0x5000 0000 in secure view.

According to the programming of the SAU and IDAU, the secure code accesses the peripheral in the secure view by generating secure transactions, and the nonsecure code accesses the same peripheral at another address in the nonsecure view. The access is either authorized or denied depending on how the peripheral security attribute is defined by GTZC/TZSC. For more details, refer to Section 4 and Section 5.

SAU configuration in the STM32CubeL5, STM32CubeU5 and STM32CubeU3

The definition of the SAU regions is made in the device partition files located in the following folders:

- STM32L5: `Drivers\CMSIS\Device\ST\STM32L5xx\Include\Templates`
- STM32U5: `Drivers\CMSIS\Device\ST\STM32U5xx\Include\Templates`
- STM32U3: `Drivers\CMSIS\Device\ST\STM32U3xx\Include\Templates`

The secure project enables the SAU and defines the SAU regions. The STM32CubeL5, STM32CubeU5, and STM32CubeU3 define the default SAU regions listed in the table below (associated with linker memory layout file templates).

Table 4. STM32CubeL5 and STM32CubeU5 default SAU regions

SAU region	STM32L5 address	STM32U5 address	STM32Cube SAU
SAU region 0	0x0C03 E000 - 0x0C03 FFFF	0x0C0F E000 - 0x0C0F FFFF	Secure, nonsecure callable
SAU region 1	0x0804 0000 - 0x0807 FFFF (256-Kbyte Flash Bank2)	0x0810 0000 - 0x081F FFFF (1 Mbyte Flash Bank2)	Nonsecure
SAU region 2	0x2001 8000-0x2003 FFFF (SRAM, 160-Kbyte second half of SRAM1 + SRAM2)	0x2004 0000 - 0x200B FFFF (SRAM3)	
SAU region 3	0x4000 0000 - 0x4FFF FFFF (Peripheral mapped memory)		
SAU region 4	0x6000 0000 - 0x9FFF FFFF (External memories)		
SAU region 5	0x0BF9 0000 - 0x0BFA 8FFF (System memory)		
SAU region 6	Not used		
SAU region 7			

Table 5. STM32CubeU3 default SAU regions

SAU region	STM32U3 address	STM32Cube SAU security attribute
SAU region 0	0x0000 0000 – 0x080F FFFF	Nonsecure
SAU region 1	0x0BF8 0000- 0x0BFA FFFF	Nonsecure
SAU region 2	0x0C07 E000 – 0X0C07 FFFF	Secure, nonsecure callable
SAU region 3	0x1000 0000 - 0x17FF FFFF	Nonsecure
SAU region 4	0x2001 8000 - 0x2002 FFFF (96 Kbyte second half of SRAM1)	Nonsecure
SAU region 5	0x2003 8000 – 0x2003FFFF (32 Kbyte second half of SRAM2)	Nonsecure
SAU region 6	0x4000 0000 – 0x4FFF FFFF (Peripheral mapped memory)	Nonsecure
SAU region 7	0x9000 0000 – 0x9FFF FFFF	Nonsecure

All memory space in 0x0000 0000-0xDFFF FFFF not covered by an SAU region is fixed as secure.

The result of the combination between the security attribute provided by the IDAU and the security attribute provided by the SAU, is shown in the tables below.

Table 6. STM32CubeL5 memory security partitioning

Region	Address range	Security attribute through IDAU	Security attribute through SAU	Final security attribute
Flash memory	0x0804 0000 - 0x0807 FFFF	Nonsecure	Nonsecure	Nonsecure
	0x0C00 0000 - 0x0C03 DFFF	Nonsecure callable	Secure	Secure
	0x0C03 E000 - 0x0C03 FFFF	Nonsecure callable	Nonsecure callable	Nonsecure callable
SRAM1	0x3000 0000 - 0x3001 7FFF	Nonsecure callable	Secure	Secure
	0x2001 8000 - 0x2002 FFFF	Nonsecure	Nonsecure	Nonsecure
SRAM2	0x2003 0000 - 0x2003 FFFF	Nonsecure	Nonsecure	Nonsecure
Peripherals	0x4000 0000 - 0x4FFF FFFF	Nonsecure	Nonsecure	Nonsecure
	0x5000 0000 - 0x5FFF FFFF	Nonsecure callable	Secure	Secure

Region	Address range	Security attribute through IDAU	Security attribute through SAU	Final security attribute
External memories	0x6000 0000 - 0x9FFF FFFF	Nonsecure	Nonsecure	Nonsecure

Table 7. STM32CubeU5 memory security partitioning

This table illustrates security partitioning applicable for STM32U575xx/U585xx devices and does not apply to STM32U535xxU545xx, STM32U99xx/U5A9xx, and STM2U5Fxx/U5Gxx devices.

Region	Address range	Security attribute through IDAU	Security attribute through SAU	Final security attribute
Flash memory	0x0810 0000 - 0x081F FFFF	Nonsecure	Nonsecure	Nonsecure
	0x0C00 0000 - 0x0C0F DFFF	Nonsecure callable	Secure	Secure
	0x0C0F E000 - 0x0C0F FFFF	Nonsecure callable	Nonsecure callable	Nonsecure callable
SRAM1	0x3000 0000 - 0x3002 7FFF	Nonsecure callable	Secure	Secure
SRAM2	0x3003 0000 - 0x3003 FFFF	Nonsecure callable	Secure	Secure
SRAM3	0x2004 0000 - 0x200B FFFF	Nonsecure	Nonsecure	Nonsecure
SRAM4	0x2800 0000-0x2800 3FFF	Nonsecure	Nonsecure	Nonsecure
Peripherals	0x4000 0000 - 0x4FFF FFFF	Nonsecure	Nonsecure	Nonsecure
	0x5000 0000 - 0x5FFF FFFF	Nonsecure callable	Secure	Secure
External memories	0x6000 0000 - 0x9FFF FFFF	Nonsecure	Nonsecure	Nonsecure

Table 8. STM32CubeU3 memory security partitioning

Region	Address range	Security attribute through IDAU	Security attribute through SAU	Final security attribute
Flash memory	0x0800 0000 - 0x080F FFFF	Nonsecure	Nonsecure	Nonsecure
	0x0C00 0000 - 0x0C07 DFFF	Nonsecure callable	Secure	Secure
	0x0C07 E000 - 0x0C0F FFFF	Nonsecure callable	Nonsecure callable	Nonsecure callable
SRAM1	0x3000 0000 - 0x3002 FFFF	Nonsecure callable	Secure	Secure
SRAM2	0x3003 0000 - 0x3003 FFFF	Nonsecure callable	Secure	Secure
Peripherals	0x4000 0000 - 0x4FFF FFFF	Nonsecure	Nonsecure	Nonsecure
	0x5000 0000 - 0x5FFF FFFF	Nonsecure callable	Secure	Secure
External memories	0x9000 0000 - 0x9FFF FFFF	Nonsecure	Nonsecure	Nonsecure

This is of course an example. The user must adapt the memory partitioning based on the application requirements in terms of secure and nonsecure resources.

4 Security configurations on STM32L5, STM32U5, and STM32U3 series

The SAU/IDAU settings are only applicable to one master: the CPU. The other masters (such as DMA) do not see these policies. That is why a local secure gate is needed on the peripheral side.

In addition to the Cortex-M33 TrustZone feature, the STM32L5/U5/U3 devices come with complementary security features that reinforce and allow a more flexible partition between the secure and the nonsecure worlds, by providing a second level of security on top of the SAU/IDAU.

4.1 Security configuration of the flash memory

The flash memory regions are configurable as secure, thanks to the nonvolatile flash secure watermark and volatile block-based flash interface registers, even if they are nonsecure through IDAU/SAU.

SAU and IDAU are responsible to grant the transactions issued by the CPU and tag the CPU access to the interconnect as secure or nonsecure. The flash memory secure watermarks and block-based registers grant the transactions from the CPU/Cortex-M33 and other masters such as:

- STM32L5 masters: DMA1, DMA2 and SDMMC
- STM32U5 masters: GPDMA1 (general purpose DMA featuring two master ports), DMA2D, SDMMC1 and SDMMC2
- STM32U3 masters: GPDMA1 (general purpose DMA featuring two master ports) and SDMMC1

As seen in Figure 2, each transaction, issued by the Cortex-M33 that targets flash memory, is first checked by IDAU/SAU, then checked by flash secure watermark or block-based registers. Refer to Figure 5 for more details.

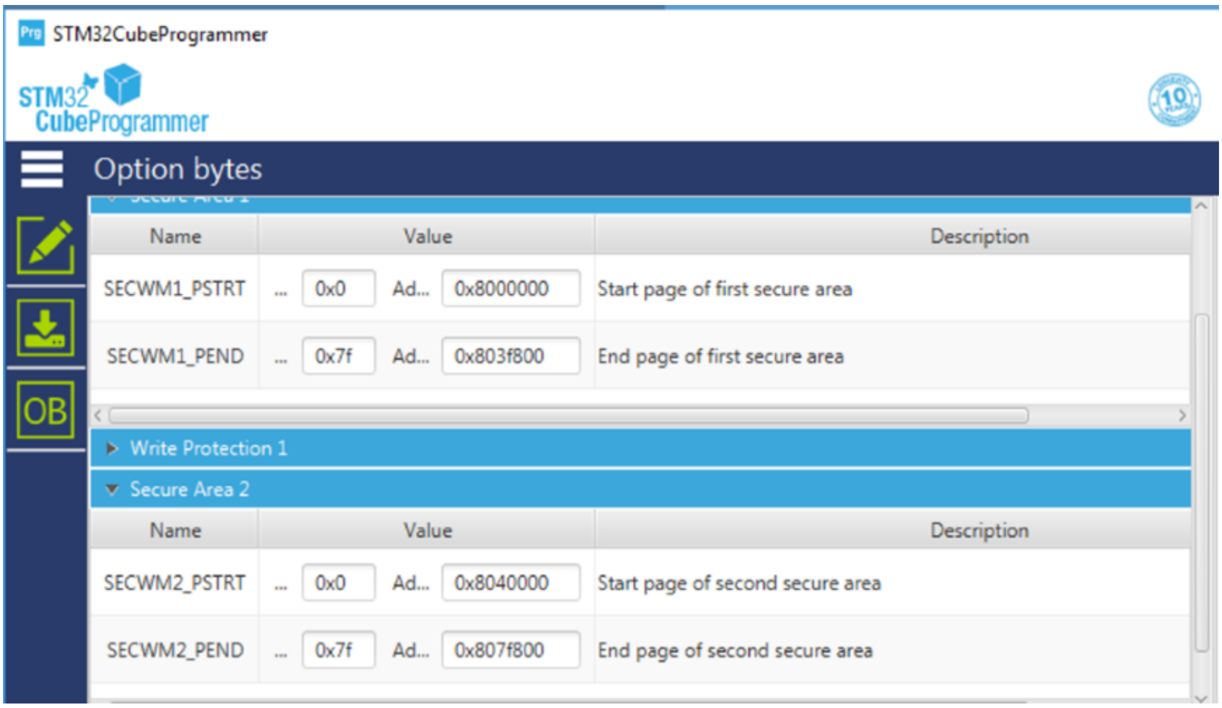
4.1.1 Secure watermark of the flash memory

The option byte defines up to two different non-volatile secure areas, and are read or written by a secure access only:

- For STM32L5/U5: SECWMx_PSTRT and SECWMx_PEND (x = 1, 2)
- For STM32U3: SECWMx_STRT and SECWMx_END (x=1, 2)

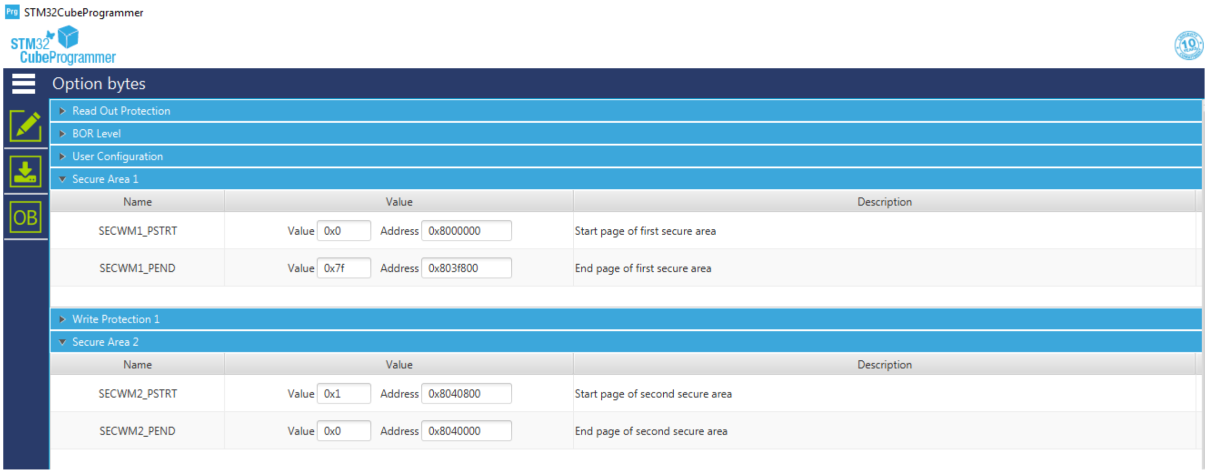
The figure below shows the default value after setting TZEN (the whole flash memory is secure).

Figure 3. **Default flash memory state though the option bytes after setting TZEN**



The STM32CubeL5, STM32CubeU5, and STM32CubeU3 TrustZone examples assume that Bank 1 is secure and Bank 2 is nonsecure.

Figure 4. Default flash bank security state defined by STM32Cube though the option bytes



4.1.2 Flash memory block-based feature

Even if the whole flash memory is nonsecure through IDAU/SAU and through the flash secure watermark option bytes, it is possible to configure temporary secure areas using the flash memory block-based feature.

Any page is programmed either in secure or nonsecure mode, using the flash interface block-based configuration registers.

Block-based registers can only set a page as secure whereas it is set as nonsecure through flash secure watermark option bytes. The opposite is not possible: a page cannot be configured as nonsecure using block-based registers, when it is configured as secure through flash secure watermark option bytes.

4.2 Global TrustZone controller (GTZC)

The GTZC has the following subblocks:

- TZSC (TrustZone security controller) allows the security attribute configuration of:
  - peripherals (see the note below) as either secure or nonsecure
  - external memories: through watermark-memory-protection controller (MPCWMx, x = 1, 2, 3) (for STM32L5/U5 only)
- MPCBBx (block-based memory protection controller) allows the security attribute configuration of SRAM blocks as follows:
  - STM32L5: The SRAM1 and SRAM2 can be programmed as secure or nonsecure by blocks, using the MPCBBx. The granularity of SRAM secure block-based is a page of 256 bytes.
  - STM32U5: The SRAM1, SRAM2, SRAM3, SRAM4 can be programmed as secure or nonsecure by blocks, using the MPCBBx. The granularity of SRAM secure block-based is a page of 512 bytes.
  - STM32U3 The SRAM1 and SRAM2 can be programmed as secure or nonsecure by blocks, using the MPCBBx. The granularity of SRAM secure block-based is a page of 512 bytes.
- TZIC (TrustZone illegal access controller) gathers all illegal access events in the system, and generates a secure interrupt towards the NVIC (GTZC_IRQn).

Note: When the TrustZone security is active, a peripheral is either securable or TrustZone-aware:

- *Securable: the security attribute is configured by GTZC/TZSC controller.*
- *TrustZone-aware: the security attribute is configured using some peripheral secure registers. For example, the GPIO is TrustZone-aware with a security attribute configured through the GPIOx_SECCFGR secure register.*

For the list of securable and TrustZone-aware peripherals, refer to 'TrustZone peripheral classification' section in the document [1], [2], or [3].

5 Overall system security access rules

5.1 Default security status

When the TrustZone security is activated by the TZEN option bit in FLASH_OPTR, the default system security state is as follows:

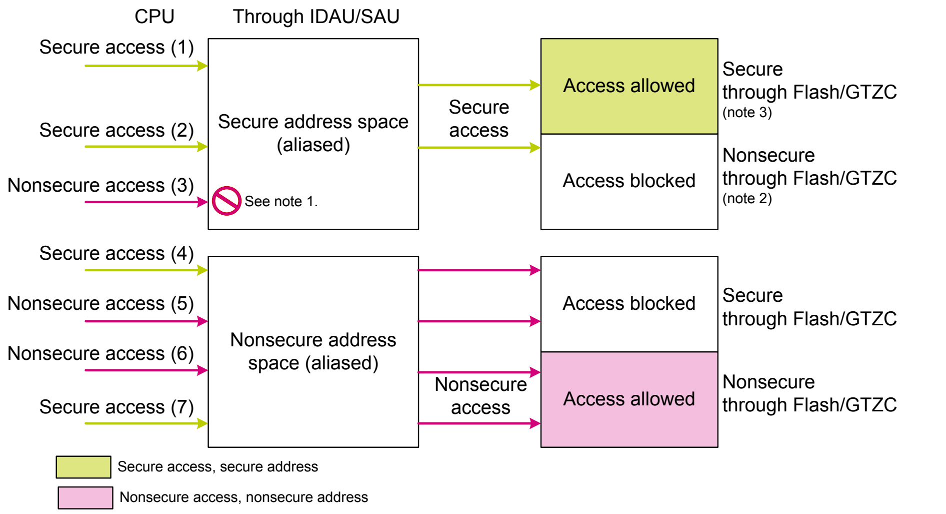
- Cortex-M33 CPU is in secure state after reset. The boot address must point toward a secure memory area.
- All interrupts are assigned to secure interrupt controller.
- All memory map is fully secure through the IDAU/SAU, until the secure code enables the SAU and define regions for nonsecure resources.
- The whole flash memory is secure. The security area is defined by watermark user option bytes, for which production values are:
  - For STM32L5/U5:
    - SECWMx_PSTRT (x = 1, 2)= 0x00
    - SECWMx_PEND (x = 1, 2) = 0x7F
  - For STM32U3:
    - SECWMX_STRT (x = 1, 2) = 0x00
    - SECWMx_END (x = 1, 2) = 0x7F
- All SRAMs are secure.
- For STM32L5/U5: External memories FSMC and OCTOSPIx banks are secure.
- For STM32U5, the backup SRAM is secure.
- All peripherals (except GPIOs) are nonsecure.
- All GPIOs are secure.
- All DMA channels are nonsecure.
- Backup registers are nonsecure.

5.2 Memory and peripheral security access rules

Any transaction issued by the CPU is filtered first by the SAU, then by the secure gate implemented close to the peripheral target (flash memory, SRAM, external memory or any secure peripheral) .

The figure below describes the transaction filtering according to transaction security attribute.

Figure 5. Memory and peripheral data access rules summary



Notes: 1. An access blocked by SAU/IDAU results in secure fault.
2. Nonsecure registers of peripherals are accessible by secure transactions as well. This is a difference with the memory access rules. Also, when the SRWILADIS bit in GTZC_MPCBBx_CR register is set, secure read/write access is allowed on nonsecure SRAM block.
3. Secure/nonsecure through Flash/GTZC refers to security attribute through flash secure watermark option bytes, flash memory block-based registers, MPCWMx, MPCBBx and TZSC_SECCFGRx registers.

The transactions carry their secure attributes. According to these attributes, the access is granted or not by the SAU, then by the flash memory or the GTZC (for SRAMs, external memories and peripherals).

Note: For STM32L5 only, the nonsecure information block is only accessible by nonsecure transactions. The information block is a flash memory region composed of option bytes, memory protection user configuration, system memory and OTP (one-time programmable) area. In particular, the OTP area, VREFINT and temperature sensor calibration values are only accessed by nonsecure transactions. The secure application must then program an SAU region configuring this area as nonsecure.

The access rules are listed below:

- Secure access to an address that is secure through SAU/IDAU and secure through Flash/GTZC: the access is allowed. See (1) in Figure 5.
- Secure access to an address that is secure through SAU/IDAU and nonsecure through Flash/GTZC: the access is blocked. See (2) in Figure 5.
- Nonsecure access to an address that is secure through SAU/IDAU: the access is blocked whatever the security attribute of the address through Flash/GTZC. A Cortex-M33 secure fault exception is triggered. See (3) in Figure 5.
- Secure access to an address that is nonsecure through SAU/IDAU and secure through Flash/GTZC: the access is blocked. See (4) in Figure 5.
- Nonsecure access to an address that is nonsecure through SAU/IDAU and secure through Flash/GTZC: the access is blocked. See (5) in Figure 5.
- Nonsecure access to an address that is nonsecure through SAU/IDAU and nonsecure through Flash/GTZC: the access is allowed. See (6) in Figure 5.
- Secure access to an address that is nonsecure through SAU/IDAU and nonsecure through Flash/GTZC: the access is allowed. See (7) in Figure 5.

When the access is blocked, the result is one of the following:

- RAZ/WI (read at zero/write ignore)
- RAZ/WI and illegal access event/interrupt
- Bus error

For example, a nonsecure access to a secure flash memory area is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the illegal access interrupt is enabled by FLASHIE in GTZC_TZIC_IER2 for STM32L5 and GTZC1_TZIC_IER4 for STM32U5/U3.

For detailed information, refer to the document [1], [2], or [3].

For an instruction fetch, the transaction output from SAU (secure or nonsecure) depends on the target address independently of the CPU state.

Table 9. Instruction fetch rules

CPU state	Target memory address security attribute through IDAU/SAU	Transaction
Secure or nonsecure	Nonsecure	Nonsecure
Secure or nonsecure	Nonsecure callable	Secure
Secure or nonsecure	Secure	Secure

6 Boot and root secure services (RSS)

The RSS is embedded in the secure information block, part of the secure flash memory area and is programmed during ST production. For more details, refer to the document [1], [2], or [3].

The RSS enables for example the secure firmware installation (SFI) thanks to the RSS extension firmware (RSSe SFI). This feature allows the customers to protect the confidentiality of the firmware to be provisioned into the STM32 device, when the production is subcontracted to a non-trusted third party. Refer to the application note *Overview secure firmware install (SFI)* (AN4992) for more details.

The boot memory address is programmed through the SECBOOTADD0[24:0] option bytes for STM32L5/U5 and the ADD[24:0] option bytes in SBOOT0R register for STM32U3. However, the allowed address space depends on the readout protection (RDP) level of the flash memory. If the programmed boot memory address is out of the allowed memory mapped area when RDP level is 0.5 or more, the default boot fetch address is forced in secure system flash memory.

Table 10. Boot space versus RDP protection

RDP level	Boot address
0	Any boot address
0.5	Only RSS or secure flash memory
1	
2	

When TrustZone is enabled by setting the TZEN option bit, the boot space must be in a secure area.

For STM32L5/U5, SECBOOTADD0[24:0] option bytes are used to select the boot secure memory address. To increase the security and establish a root of trust (RoT), a unique boot entry option must be selected regardless the other boot options. This is done by setting the BOOT_LOCK option bit in the FLASH_SECBOOTADD0R register. This bit must only be set by a secure access.

For STM32U3, ADD[24:0] option bytes in the SBOOT0R register are used to select the boot secure memory address. To increase the security and establish a root of trust (RoT), a unique boot entry option must be selected regardless the other boot options. This is done by setting the BOOT_LOCK option bit in the FLASH_SBOOT0R register. This bit must only be set by a secure access.

Caution: *For STM32L5, the BOOT_LOCK option bit, once set, cannot be cleared. The unique boot entry address is the address programmed in SECBOOTADD0[24:0] option bytes. For STM32U5/U3, BOOT_LOCK can only be cleared in RDP level 0.*

7 Readout protection (RDP) when TrustZone enabled

When TrustZone is enabled (TZEN = 1), the four RDP levels, from no protection (level 0) to maximum protection with no debug (level 2), are detailed in the table below.

Table 11. RDP protection levels (when TrustZone enabled)

RDP byte value	RDP level
0xAA	0
0x55	0.5
Any value except 0x55, 0xAA, or 0xCC	1
0xCC	2

7.1 RDP level 1

In RDP level 1, the flash main memory, the backup registers, the backup RAM (STM32U5 only), OTFDEC region (when available), ICACHE, DCACHE, and SRAMs cannot be accessed. An intrusion is detected in case of debug access when the CPU is in secure state.

When the CPU is in nonsecure state, connections to the target through JTAG/SWD and RDP regression are possible. Any debug access other than halting the CPU is considered as an intrusion.

The RDP regression must be done in one of the following ways:

- Through bootloader: the boot must be done from the RSS.

Note: With a boot from RSS, it is possible to do regression through JTAG/SWD as well.

- Through JTAG/SWD, with a boot from the user flash memory: The CPU must be in nonsecure state to be able to connect to the target.
  Before programming the RDP level 1, the user must ensure that the secure application calls the non-secure application (possible connection to the target), and that the regression from RDP level 1 is possible.

Caution: *If there is no nonsecure code, the CPU always remains in the secure state and the RDP regression cannot be done through JTAG/SWD with a boot from the user flash memory. In this case, the only way to do regression is through JTAG/SWD/bootloader with a boot from RSS. For more details, refer to the 'Boot configuration' section of the document [1] or [2]. In STM32U5/U3, the RDP level 1 regression can be protected by OEM1 and OEM2 keys that have been provisioned at lower RDP levels (refer to document [2] or [3] for more details).*

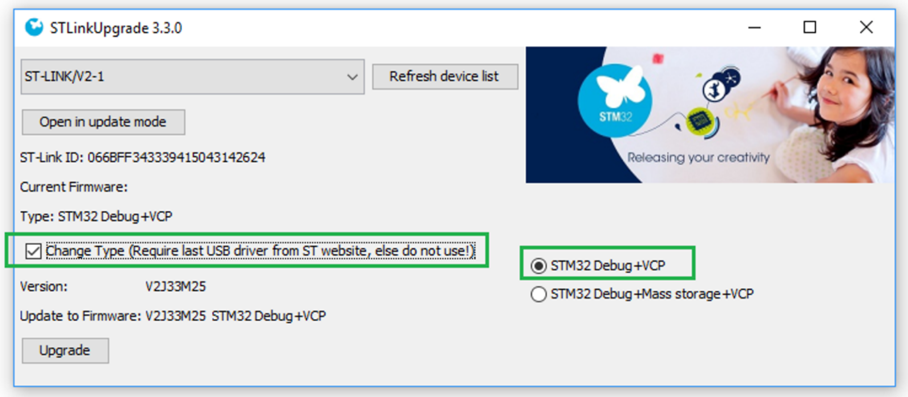
The ST boards for STM32L5/U5/U3 (Nucleo, Evaluation and Discovery kit) come with the integrated ST-LINK that can be used as power supply and for debug at the same time. Due to the mass storage interface of the ST-LINK that requires the target identification at ST-LINK startup (SWD connection), a debug intrusion is detected every time the ST-LINK USB cable is plugged. The user application is never executed and the CPU remains always in LOCKUP state and no code execution is possible in case of an intrusion. The target cannot then be connected.

The following solutions must be used to maintain the user application execution and to connect to the target through JTAG/SWD:

- Use the ST-LINK for debug only: Another supply source must be used. To execute the user application, a power off/on must be applied while the ST-LINK USB cable is already plugged.
- Disable the mass storage interface of the ST-LINK, by changing the firmware type through the STLinkUpgrade application, as shown in Figure 6.

Note: The ability to disable the mass storage has only been implemented in ST-Link/V2.

Figure 6. Disabling the mass storage interface of the ST-LINK



7.2 RDP level 0.5

In RDP level 0.5, the secure flash memory, the secure backup registers, the backup RAM (STM32U5 only), OTFDEC region (when available), ICACHE, DCACHE, and SRAMs area cannot be accessed. The nonsecure flash memory, the nonsecure backup registers and the nonsecure SRAMs area remain accessible.

When the CPU is in secure state, it is not possible to connect to the target through JTAG/SWD.

When the CPU is in nonsecure state, connection to the target through JTAG/SWD and RDP regression are possible.

Note: In STM32U5/U3, at RDP level 0.5, it is not possible to request RDP level 0. An RDP increase to level 1 followed by a RDP regression to level 0 is required.

The RDP regression is done in one of the following ways:

- through bootloader: the boot must be done from the RSS.

  Note: With a boot from RSS, it is possible to do regression through JTAG/SWD as well.

- Through JTAG/SWD, with a boot from the user flash memory: In order to be able to connect to the target, the CPU must be in nonsecure state because the secure debug is forbidden in RDP level 0.5. It is then possible to connect to the target only when the CPU is in nonsecure state.
  Before programming the RDP level 0.5, the user must always ensure that the secure application calls the nonsecure application (possible connection to the target), and that the regression from RDP level 0.5 is possible.

  Caution: *If there is no nonsecure code, the CPU always remains in the secure state and the RDP regression cannot be done through JTAG/SWD with a boot from the user flash memory. In this case, the only way to do regression is through JTAG/SWD/bootloader with a boot from RSS. For more details, refer to the 'Boot configuration' section of the document [1] or [2].*

For more details about the different readout protection levels and the access status versus protection level and execution mode when TZEN = 1, refer to the document [1] or [2].

Note: In RDP level 1 and 0.5, when the STM32CubeProgrammer is used, the connection to the target must be done in "Hot plug" mode, in order to keep the user application executing during the connection to the target and to avoid a connection when the CPU is in reset state (means CPU secure).

Caution: *It is not possible to do RDP regression when the following conditions are met:*

- *BOOT_LOCK option bit is set.*
- *For STM32L5/U5: SECBOOTADD0[24:0] is an address in the secure user flash memory.*
- *For STM32U3: ADD[24:0] in the FLASH_SBOOT0R register is an address in the secure user flash memory.*
- *There is no nonsecure code. The CPU is always in secure state and it is not possible to program the nonsecure flash memory in RDP level 0.5.*

7.3 RDP level 2

When the RDP level 2 is set, the protection level 1 is guaranteed. The boot from SRAM (boot RAM mode) and the boot from system memory (bootloader mode) are no longer available. Only boot from main flash memory or RSS are possible.

When booting from main flash memory or RSS, all operations are allowed on the main flash memory. Read, erase and program accesses to flash memory and SRAMs from user code are allowed.

For STM32U5/U3, unless an OEM2 key has been provisioned, the following features are applicable:

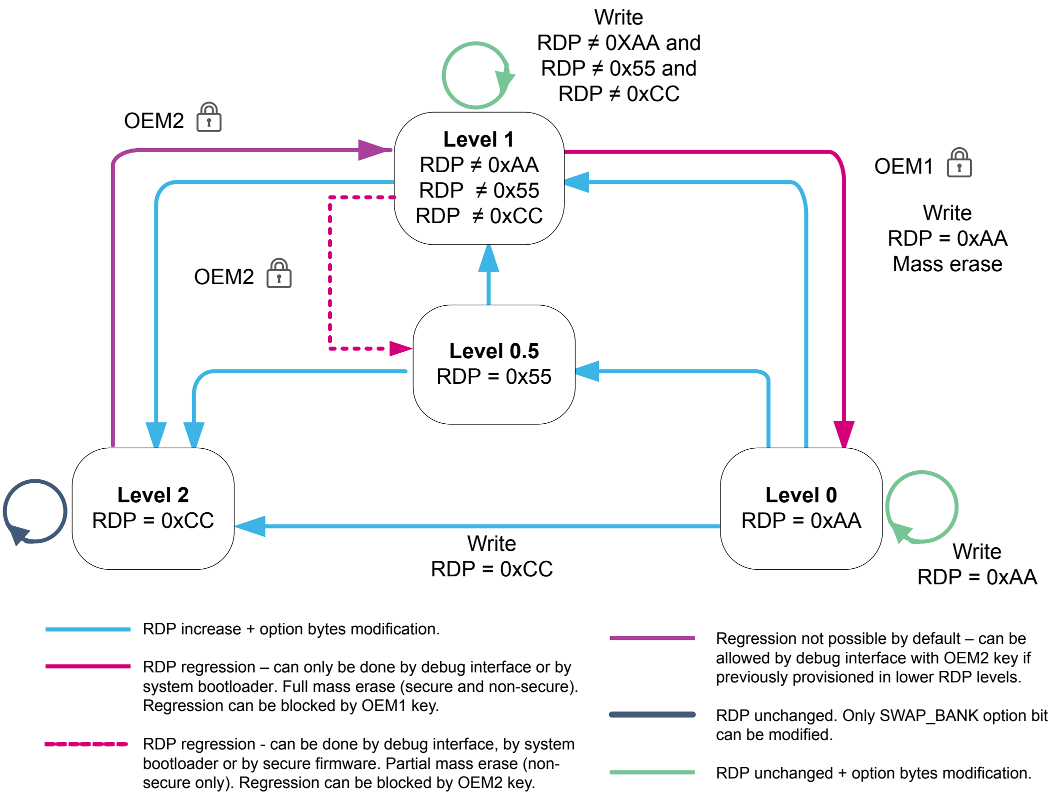
- Option bytes cannot be programmed nor erased except the SWAP_BANK option bit.
- The RDP level 2 cannot be removed (irreversible operation).
- All debug features are disabled. Debug is also disabled under reset.
- JTAG and SWD are definitively disabled. The regression is possible using the JTAG/SWD under reset mode.

Note: In STM32U5/U3, if an OEM2 key has been provided under lower RDP protection, JTAG and SWD remain enabled under reset only to interface with DBGMCU_SR, DBGMCU_DBG_AUTH_HOST and DBGMCU_DBG_AUTH_DEVICE registers, to obtain the device identification and to provide this OEM2 key to request RDP regression.

7.4 RDP transitions with OEM keys (STM32U5/U3)

The figure below shows the RDP level transition scheme when TrustZone is enabled (TZEN = 1).

Figure 7. RDP level transition scheme when TrustZone is enabled



Two keys (OEM1KEY and OEM2KEY) can be defined to lock the RDP regression (available with or without TrustZone).

- For STM32U5, OEM1KEY and OEM2KEY are 64-bit keys.
- For STM32U3, OEM1KEY and OEM2KEY are 128-bit keys.

Without OEM2 lock mechanism, the transition to RDP level 2 is irreversible and causes the device to be completely locked. This is why it is recommended to activate the OEM2 lock mechanism to allow the possibility to do a regression from RDP level 2.

- OEM1KEY can be modified:
  - in RDP level 0
  - in RDP level 0.5 or level 1 if OEM1LOCK bit is cleared
- OEM2KEY can be modified:
  - in RDP level 0 or level 0.5
  - in RDP level 1 if OEM2LOCK bit is cleared

For STM32U5, to perform a regression, shift OEMxKEY[31:0], then OEMxKEY[63:32] through JTAG or SWD in the DBGMCU_DBG_AUTH_HOST register. If key matches OEMxKEY, the RDP regression is launched by hardware.

For STM32U3, to perform a regression, shift OEMxKEY[31:0], OEMxKEY[63:32], OEMxKEY[95:64], and then OEMxKEY[127:96] through JTAG or SWD in the DBGMCU_DBG_AUTH_HOST register. If key matches OEMxKEY, the RDP regression is launched by hardware.

Note: *For STM32U5, the RDP transition using OEM keys mechanism can be disabled by writing 0xFFFFFFFF 0xFFFFFFFF in the corresponding OEM key option bytes. This can only be done when the OEM key modification is possible.*

Caution: *For STM32U3, once set, the RDP transition with OEM keys mechanism cannot be disabled. This means that the OEM keys can be modified, but never cleared.*

Refer to Section 10 and documents [2] and [3] for more details.

8 Security features available only when TrustZone is enabled

The following features are available only when TrustZone is enabled:

- GTZC secure watermark protection
- HDP (hide protection) option bytes
- Flash memory block-based secure protection
- RDP level 0.5
- RSS and SFI
- BOOT_LOCK
- Secure interrupts
- GTZC secure protection

9 TrustZone deactivation

As mentioned in Section 2.1, the TrustZone is disabled by default in all STM32L5/U5/U3 devices. The TrustZone is activated by setting the TZEN option bit.

The TrustZone deactivation must be done in parallel to an RDP regression (see Section 7.2). This assumes that the system is already in RDP level 1 or RDP level 0.5 (regression from level 0.5 is applicable only to STM32L5). See Section 7.1 and Section 7.2 for the associated recommendations to take into account.

After the TrustZone deactivation, all features mentioned in Section 8 are no longer available and all secure registers are RAZ/WI. The GTZC can still be used to configure the privilege access.

Following a regression from TZEN = 1 to TZEN = 0, the sample is virgin, corresponding to the production state.

Note: For STM32L5 only, if the BOOT_LOCK option bit is set, it cannot be cleared. After clearing TZEN and setting it again, the BOOT_LOCK remains set and the unique boot entry address is the address programmed in SECBOOTADD0[24:0] option bytes.

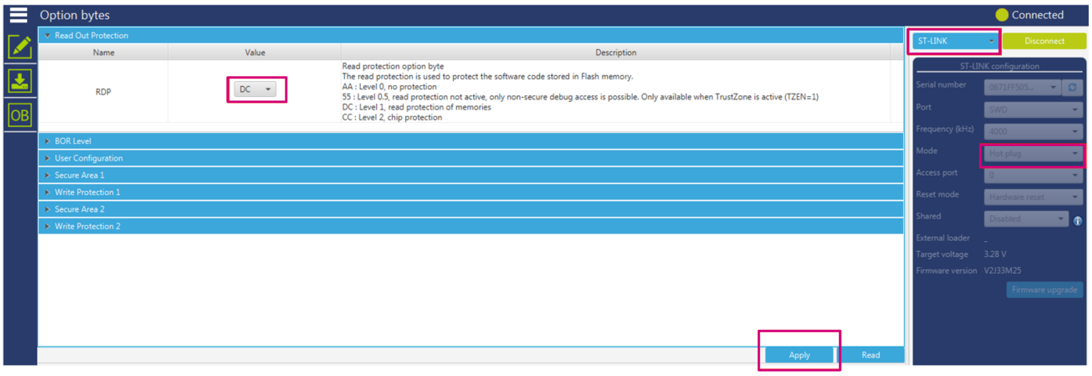
9.1 TrustZone/RDP deactivation demonstration using the STM32CubeProgrammer

9.1.1 TZEN/RDP regression with a boot from user flash memory

The following sequence is needed to perform a TZEN and RDP regression with a boot from user flash memory.

1. Make sure that secure and nonsecure applications are well loaded and executed.
2. Set RDP to level 1 (or level 0.5 for STM32L5) through STM32CubeProgrammer (intrusion occurs). Then only 'Hot Plug' connection is possible.

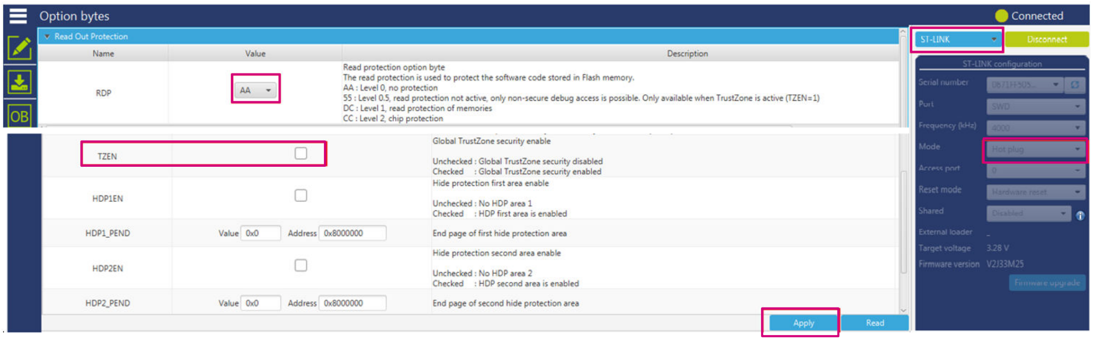
Figure 8. Setting RDP to level 1



3. Choose one of the following alternatives to recover from intrusion:
   a. Use a power supply different from ST-LINK (more details in Section 7.1) in order to be able to connect to the target.
   b. Remove the IDD jumper, then put it back in place to exit from intrusion.

4. Set RDP to level 0 (option byte value 0xAA) and uncheck the *TZEN* box, then click on *Apply*.

Figure 9. TZEN and RDP regression through SWD with boot from user Flash



If the TZEN and RDP regression with a boot from user flash, was not successful due to the fact that the first step was not respected (secure application not calling a nonsecure application), the only way to do a regression is with a boot from RSS as shown in the next section.

9.1.2 TZEN/RDP regression with a boot from RSS

This section explains how to change the boot on STM32L5/U5/U3 ST boards.

The boot must be done from RSS by applying a high level on BOOT0 pin:

- On an evaluation board (STM32L552E-EV or STM32U575I-EV), a switch SW1 is provided to change the boot (see the document [5] or [6]).
- On a Discovery kit (STM32L562E-DK or B-U585I-IOT02A), a small rework must be done to change the boot from RSS (see the document [7] or [8]).
- On a Nucleo board (NUCLEO-L552ZE-Q or NUCLEO-U575ZI-Q or NUCLEO-U385RG-Q), a connection between CN11 pin 5 (VDD) and pin 7 (PH3_BOOT0) must be done (see the document [9] or [10]).

The following sequence is recommended to boot from RSS :

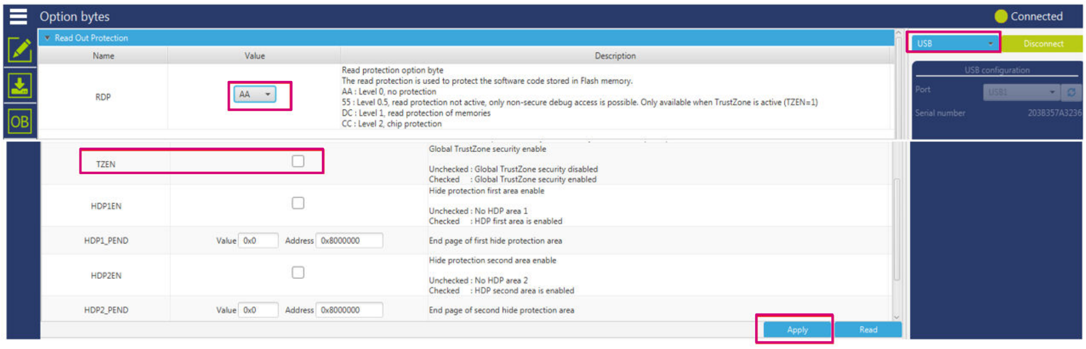
1. Make sure the following actions are completed:
   a. nSWBOOT0 option byte is checked (BOOT0 taken from PH3/BOOT0 pin)
   b. High-level voltage is applied on PH3/BOOT0 pin.
   c. For STM32L5/U5: NSBOOTADD1 option byte is configured to 0x17F200 value at 0x0BF9 0000 address (RSS address).
      For STM32U3: ADD[24:0] option byte in FLASH_BOOT1R is configured to 0x17F1E0 value at 0x0BF8 F000 address (RSS address).
   d. BOOT_LOCK option byte is unchecked (boot based on the pad/option bit configuration).
2. Set RDP to level 1 through STM32CubeProgrammer (Intrusion occurs). Then only 'Hot plug' connection is possible.
3. Recover from intrusion with one of the following alternatives:
   a. Remove the IDD jumper, then put it back in place to exit from intrusion.
   b. Use a power supply different from ST-LINK in order to be able to connect to the target.
4. Set RDP to level 0 (option byte value 0xAA) and uncheck the TZEN box. Then click on *Apply*.

The regression can be done through JTAG/SWD or bootloader as detailed below:

- Through JTAG/SWD, set RDP to level 0 (option byte value 0xAA) and uncheck the *TZEN* box, then click on *Apply*.
- Through bootloader, using one of the supported communication interfaces (USB in this example):
  – If RDP is set to level 0.5, set RDP level to level 0 (option byte value 0xAA) and uncheck the *TZEN* box, then click on *Apply*.

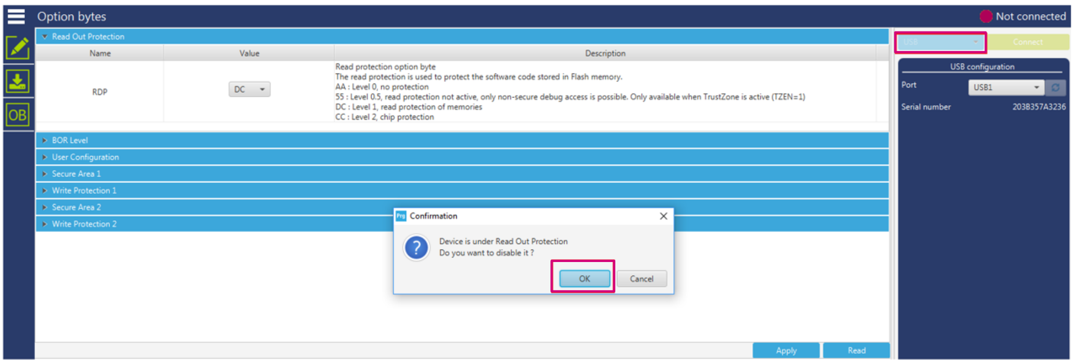
Note: Regression from RDP level 0.5 to level 0 is possible only with STM32L5. For STM32U5/U3, the RDP level must be first raised from level 0.5 to level 1, then revert from level 1 to level 0, in parallel with TZEN deactivation.

Figure 10. TZEN and RDP regression (level 0.5 to level 0) through bootloader



  – If RDP is set to level 1, the RDP regression with the STM32CubeProgrammer graphical interface is shown in the figure below.

Figure 11. RDP regression (level 1 to level 0) through bootloader



If RDP is set to level 1 and the TZEN regression cannot be done using the STM32CubeProgrammer graphical interface, the STM32CubeProgrammer CLI (command line instructions) must be used, applying the following TZEN regression command:

```
> STM32_Programmer_CLI.exe -c port=USB1 -tzenreg
      -------------------------------------------------------------------
                        STM32CubeProgrammer v2.8.0
      -------------------------------------------------------------------
USB speed    : Full Speed (12MBit/s)
Manuf. ID    : STMicroelectronics
Product ID   : DFU in FS Mode
SN           : 207E31953536
FW version   : 0x011a
Device ID    : 0x0482
Warning: Device is under Read Out Protection
Disabling TrustZone...
Disabling TrustZone successfully
```

9.1.3 TZEN/RDP regression when RDP level 1 is locked by an OEM1 Key

This feature is available in STM32U5/U3. The OEM1 RDP lock mechanism is active when the OEM1LOCK bit is set. It blocks the RDP level 1 to RDP level 0 regression.

For STM32U5, locking the RDP level 1 with an OEM1 key can be done with the following CLI command in an example where OEM1Key [31:0] = 0xABCDEFAB and OEM1Key [63:32] = 0x12345678.

```
>STM32_Programmer_CLI.exe -c port=swd  mode=hotplug  -lockRDP1 0xABCDEFAB 0x12345678
      -------------------------------------------------------------------
                        STM32CubeProgrammer v2.8.0
      -------------------------------------------------------------------

    Lock RDP1 password successfully done
```

To raise the RDP level to 1, the following CLI can be applied.

```
>STM32_Programmer_CLI.exe -c port=swd  mode=hotplug  -ob rdp=0xDC
      -------------------------------------------------------------------
                        STM32CubeProgrammer v2.8.0
      -------------------------------------------------------------------

    Option Bytes successfully programmed
```

As explained in Section 2.1, TZEN deactivation must be done in parallel with RDP level 1. The OEM1 key must be provided to unlock the RDP level 1 regression. The following STM32CubeProgrammer CLI can be applied for TZEN + RDP level 1 to level 0 regressions with OEM1 key.

```
> STM32_Programmer_CLI.exe -c port=swd mode=UR -unlockRDP1 0xABCDEFAB 0x12345678 -ob RDP=0xAA
 TZEN=0
      -------------------------------------------------------------------
                        STM32CubeProgrammer v2.8.0
      -------------------------------------------------------------------

    Unlock RDP1 password successfully done
    Option Bytes successfully programmed
```

Note: For STM32U3, the same CLI commands can be done but with a 128-bit key. For example OEM1Key [31:0] = 0xABCDEFAB , OEM1Key[63:32] = 0x12345678, OEM1Key [95:64] = 0xABCDEFAB and OEM1Key[127:96] = 0x12345678.

Caution: *For STM32U3, once set, the OEM key lock mechanism cannot be disabled. This means that the OEM key can be modified but never cleared.*

10 Demonstration of RDP transitions using OEM keys on STM32U5

In order to reach the best protection level, it is recommended to activate TrustZone and to set the RDP level 2 with password authentication regression enabled.

The RDP protects the flash main memory, the option bytes, the backup registers, the backup RAM (STM32U5 only), OTFDEC region (when available), ICACHE, DCACHE, and SRAMs. Two keys (OEM1KEY and OEM2KEY) can be defined in order to lock the RDP regression. When TrustZone is activated, the CPU is split into secure and nonsecure zones, with a set of protections described in previous sections.

This section demonstrates how to practice the RDP level transitions using the STM32CubeProgrammer CLI when OEM1KEY and OEM2KEY are provisioned to unlock RDP regressions on STM32U5.

Note: For more details on RDP transitions with OEMxKEY, refer to the document [2] and [3].

In this STM32U5 example, when the CPU is secure, the RDP regression is done through the RSS. This can be performed by connecting the PH3_BOOT0 PIN to VDD on the board. The table below summarizes the transition sequences detailed in sections linked in the first column (when the CPU is nonsecure, all these steps are applicable except the transition to/from level 0.5).

Table 12. Demonstration steps for RDP transitions using OEMxKEYs on STM32U5

Step number and title	Description	Comments
Step 1 - Provision OEM1KEY	Provision OEM1Key to unlock RDP1 to RDP0 regression (`OEM1KEY=0x11ABCDEF 0x12ABCDEF`). .	The user can choose any 64-bit key length except all 1 or all 0. When `0xFFFFFFFF 0xFFFFFFFF` is used, the OEMxKEYs are cleared.
Step 2 - Provision OEM2KEY	Provision OEM2Key (`OEM2KEY=0x21ABCDEF 0x22ABCDEF`): • to authorize RDP2 to RDP1 regression • to unlock RDP1 to RDP0.5 regression	
Step 3 - Check if OEMxKEYs are provisioned	OEM1LOCK and OEM2LOCK bits set to 1.	If OEMxKEYs are not well provisioned, the user must repeat the failed step 1 or step 2, and recheck again
Step4 - Set option byte TZEN = 1	Set the CPU as secure.	TrustZone enabled
Step 5 - Set RDP level 2	Rise RDP to level 2 (`-ob rdp=0xCC`). This shows that, when OEM2KEY is provisioned, the regression from RDP level 2 to level 1 is authorized.	Make sure that Step 2 is passed, otherwise the device is no more accessible.
Step 6 - Unlock RDP level 2 with OEM2Key	Authorize RDP level 2 to level 1 regression. The correct OEM2KEY must be provided.	• Under-reset (UR) mode is required. • Boot from RSS is required when TZEN = 1.
Step 7 - Set RDP to level 1	Regression of RDP to level 1 is now possible (unlocked by Step 6).	If not successful, it means that RDP level 2 to level 1 regression is not unlocked with the correct OEM2KEY.
Step 8 - Unlock RDP level 1 with OEM2 key	Enable the RDP level 1 to level 0.5 regression.	Be sure that, in the CLI, the option `-unlockrdp1` is used with `OEM2KY=0x21ABCDEF 0x22ABCDEF`.
Step 9 - Set RDP level 0.5	RDP level 1 to level 0.5 regression is now possible as OEM2Key is provided in Step 8.	-
Section 10.10: Step 10 - Raise RDP from level 0.5 to level 1	-	As an RDP level 0.5 to level 0 regression is not possible, the user must first rise the RDP to level 1.
Section 10.11: Step 11 - Unlock RDP level 1 with OEM1Key	The correct OEM1KEY must be provided.	-
Step 12 - Set RDP level 0 and reset TZEN = 0	TZEN + RDP level 1 to level 0 regression	-

10.1 Step 1 - Provision OEM1KEY

Provision OEM1KEY (`0x11ABCDEF 0x12ABCDEF`) in order to lock RDP level 1 to level 0 regression. Use the following command line:

```
> STM32_Programmer_CLI.exe -c port=swd mode=hotplug -lockrdp1 0x11ABCDEF 0x12ABCDEF
Device name : STM32U575/STM32U585
Flash size  : 2 MBytes
Device type : MCU
Device CPU  : Cortex-M33
BL Version  : 0x30
Debug in Low Power mode enabled
Lock RDP1 password successfully done
```

10.2 Step 2 - Provision OEM2KEY

Provision OEM2KEY (`0x21ABCDEF 0x22ABCDEF`) in order to authorize RDP level 2 to level 1 regression, or to lock RDP level 1 to level 0.5 regression. Use the following command line:

```
> STM32_Programmer_CLI.exe -c port=swd mode=hotplug -lockrdp2 0x21ABCDEF 0x22ABCDEF
Device name : STM32U575/STM32U585
Flash size  : 2 MBytes
Device type : MCU
Device CPU  : Cortex-M33
BL Version  : 0x20
Debug in Low Power mode enabled
Lock RDP2 password successfully done
```

10.3 Step 3 - Check if OEMxKEYs are provisioned

Check that OEM2LOCK and OEM1LOCK bits in the FLASH_NSSR register are set to be sure that OEM1KEY/ OEM2KEY are provisioned and locked . Use the following command line:

```
> STM32_Programmer_CLI.exe -c port=swd mode=hotplug -r32 0x40022020 4
Reading 32-bit memory content
  Size          : 4 Bytes
  Address:      : 0x40022020

0x40022020 : 000C0000   --> FLASH_NSSR[19:18]=11
```

Using the GUI check the memory content at address 0x40022020 or the FLASH_NSSR register.

10.4 Step4 - Set option byte TZEN = 1

Program the TZEN option byte using the following command to enable the TrustZone.

```
>STM32_Programmer_CLI.exe -c port=swd mode=hotplug -ob TZEN=1

UPLOADING OPTION BYTES DATA ...
  Bank          : 0x00
  Address       : 0x50022040
  Size          : 32 Bytes
██████████████████████████████████████████████████ 100%
  Bank          : 0x01
  Address       : 0x50022060
  Size          : 8 Bytes
██████████████████████████████████████████████████ 100%
  Bank          : 0x02
  Address       : 0x50022068
  Size          : 8 Bytes
██████████████████████████████████████████████████ 100%
OPTION BYTE PROGRAMMING VERIFICATION:
Option Bytes successfully programmed
```

10.5 Step 5 - Set RDP level 2

Rise RDP to level 2 by setting the option bytes (`-ob rdp=0xCC`) with the following command:

```
>STM32_Programmer_CLI.exe -c port=swd mode=hotplug -ob rdp=0xCC


UPLOADING OPTION BYTES DATA ...
  Bank          : 0x00
  Address       : 0x50022040
  Size          : 32 Bytes
████████████████████████████████████████████████ 100%
  Bank          : 0x01
  Address       : 0x50022060
  Size          : 8 Bytes
████████████████████████████████████████████████ 100%
  Bank          : 0x02
  Address       : 0x50022068
  Size          : 8 Bytes
████████████████████████████████████████████████ 100%
PROGRAMMING OPTION BYTES AREA ...
  Bank          : 0x00
  Address       : 0x50022040
  Size          : 32 Bytes
░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░
Reconnecting...
Error: failed to reconnect after reset !
UPLOADING OPTION BYTES DATA ...
  Bank          : 0x00
  Address       : 0x40022040
  Size          : 32 Bytes
░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░
Error: Uploading Option Bytes bank: 0 failed
Error: Reloading Option Bytes Data failed  --> Not possible to reconnect as RDP2
```

10.6 Step 6 - Unlock RDP level 2 with OEM2Key

Provide OEM2KEY to authorize the transition from RDP level 2 to the RDP level 1.

Note: The under-reset mode (UR) is recommended.

Use the following command:

```
>STM32_Programmer_CLI.exe -c port=swd mode=UR -unlockrdp2 0x21ABCDEF 0x22ABCDEF
      -------------------------------------------------------------------
                       STM32CubeProgrammer v2.8.0
      -------------------------------------------------------------------
ST-LINK SN  : 0028003D3038510234333935
ST-LINK FW  : V3J8M3
Board       : NUCLEO-U575ZE
Voltage     : 3.31V
Unlock RDP2 password succefully done!
Error: Cannot connect to access port 0
If you are trying to connect to a device with TrustZone enabled please try to connect with Ho
tPlug mode
```

If some issues occur:

- Check that the system boots from the RSS and that PH3-BOOT0 PIN of the board is connected to VDD.
- Check that DBGMCU is accessible on RDP level 2 (DBGMCU_CR @0xE0044000) with the following command:

```
>STM32_Programmer_CLI.exe -c port=swd mode=hotplug -r32 0xE0044104 4
Reconnected with the recommended frequency (3300 kHz)!
Device name : STM32U575/STM32U585
Flash size  : 2 MBytes
Device type : MCU
Device CPU  : Cortex-M33
BL Version  : 0x20
Debug in Low Power mode enabled
Reading 32-bit memory content
  Size          : 4 Bytes
  Address:      : 0xE0044104

0xE0044104 : 292D8E4A
```

If DBGMCU is not accessible, it means that OEM2KEY was not provisioned.

10.7 Step 7 - Set RDP to level 1

Launch the RDP level 2 to level 1 regression (unlocked by OEM2KEY in Step 6) with the following command:

```
> STM32_Programmer_CLI.exe -c port=swd mode=hotplug -ob rdp=0xDC
….
UPLOADING OPTION BYTES DATA ...
  Bank          : 0x00
  Address       : 0x40022040
  Size          : 32 Bytes
██████████████████████████████████████████████████ 100%
  Bank          : 0x01
  Address       : 0x40022060
  Size          : 8 Bytes
██████████████████████████████████████████████████ 100%
  Bank          : 0x02
  Address       : 0x40022068
  Size          : 8 Bytes
██████████████████████████████████████████████████ 100%
OPTION BYTE PROGRAMMING VERIFICATION:
Option Bytes successfully programmed
```

10.8 Step 8 - Unlock RDP level 1 with OEM2 key

OEM2KEY is provisioned to authorize RDP level 2 to RDP level 1, and RDP level 1 to RDP level 0.5 regressions. For the second one, the RDP level 1 must be unlocked with OEM2KEY (`0x21ABCDEF 0x22ABCDEF`), using the following command:

```
> STM32_Programmer_CLI.exe -c port=swd mode=hotplug -unlockrdp1 0x21ABCDEF 0x22ABCDEF
      -------------------------------------------------------------------
                        STM32CubeProgrammer v2.8.0
      -------------------------------------------------------------------
ST-LINK SN  : 0028003D3038510234333935
ST-LINK FW  : V3J8M3
Board       : NUCLEO-U575ZE
Voltage     : 3.31V
SWD freq    : 24000 KHz
Connect mode: Hot Plug
Reset mode  : Software reset
Device ID   : 0x482
Revision ID : Rev B
Reconnecting with the recommended frequency (1000 kHz)!
…
Reconnected with the recommended frequency (3300 kHz)!
Device name : STM32U575/STM32U585
Flash size  : 2 MBytes
Device type : MCU
Device CPU  : Cortex-M33
BL Version  : 0x90
Debug in Low Power mode enabled

Unlock RDP1 password successfully done
```

10.9 Step 9 - Set RDP level 0.5

Launch the RDP level 1 to level 0.5 regression (unlocked by OEM2KEY in Step 8) with the following command:

```
>STM32_Programmer_CLI.exe -c port=swd mode=hotplug -ob rdp=0x55
….
UPLOADING OPTION BYTES DATA ...
  Bank          : 0x00
  Address       : 0x40022040
  Size          : 32 Bytes
██████████████████████████████████████████████████ 100%
  Bank          : 0x01
  Address       : 0x40022068
  Size          : 8 Bytes
██████████████████████████████████████████████████ 100%
OPTION BYTE PROGRAMMING VERIFICATION:
Option Bytes successfully programmed
```

10.10 Step 10 - Raise RDP from level 0.5 to level 1

Raise RDP to level 1 to be able to reach RDP level 0 in next step (no RDP level 0.5 to level 0 authorized), with the following command:

```
> STM32_Programmer_CLI.exe -c port=swd mode=hotplug -ob rdp=0xDC
….
UPLOADING OPTION BYTES DATA ...
  Bank          : 0x00
  Address       : 0x40022040
  Size          : 32 Bytes
██████████████████████████████████████████████████ 100%
  Bank          : 0x01
  Address       : 0x40022068
  Size          : 8 Bytes
██████████████████████████████████████████████████ 100%
OPTION BYTE PROGRAMMING VERIFICATION:
Option Bytes successfully programmed
```

10.11 Step 11 - Unlock RDP level 1 with OEM1Key

The correct OEM1KEY must be provided, otherwise no regression is possible. The following command unlocks the RDP level 1:

```
> STM32_Programmer_CLI.exe -c port=swd mode=hotplug -unlockrdp1 0x11ABCDEF 0x12ABCDEF
…
Reconnected with the recommended frequency (3300 kHz)!
Device name : STM32U575/STM32U585
Flash size  : 2 MBytes
Device type : MCU
Device CPU  : Cortex-M33
BL Version  : 0xf0
Debug in Low Power mode enabled

Unlock RDP1 password successfully done
```

10.12 Step 12 - Set RDP level 0 and reset TZEN = 0

When TrustZone is enabled, the deactivation of TZEN option byte must be done simultaneously with RDP level 1 to level 0 regression. Use the following commands:

```
> STM32_Programmer_CLI.exe -c port=swd mode=hotplug -ob rdp=0xAA tzen=0
….
UPLOADING OPTION BYTES DATA ...
  Bank          : 0x00
  Address       : 0x40022040
  Size          : 32 Bytes
████████████████████████████████████████████████ 100%
  Bank          : 0x01
  Address       : 0x40022068
  Size          : 8 Bytes
████████████████████████████████████████████████ 100%
OPTION BYTE PROGRAMMING VERIFICATION:
Option Bytes successfully programmed
```

Note: When TrustZone is disabled, the command is the following:

```
STM32_Programmer_CLI.exe -c port=swd mode=hotplug -ob rdp=0xAA
```

10.13 Clear OEMxKEYs

Clear OEM1Key and/or OEM2KEY by writing `0xFFFFFFFF 0xFFFFFFFF` in both OEM1KEY and OEM2KEY flash memory option bytes.

The OEM1_LOCK and/or OEM2_LOCK bits in FLASH_NSSR [19:18] are consequently cleared. They can be checked as described in Section 10.3 (FLASH_NSSR [19:18] = 00).

Caution: *When OEM2KEY is cleared or not provisioned at all, RDP level 2 to level 1 regression is not possible.*

When OEM1 and OEM2 keys are initially provisioned with keys as in Step 1 (Section 10.1) and Step 2 (Section 10.2):

- The OEM1key can be cleared at RDP level 0.
- The OEM2Key can be cleared at RDP level 1, level 0.5, or level 0.

Clear OEM1KEY

Use the following command in the STM32CubeProgrammer CLI interface:

```
> STM32_Programmer_CLI.exe -c port=swd mode=hotplug -lockrdp1 0xFFFFFFFF  0xFFFFFFFF
Device name : STM32U575/STM32U585
Flash size  : 2 MBytes
Device type : MCU
Device CPU  : Cortex-M33
BL Version  : 0x30
Debug in Low Power mode enabled
Lock RDP1 password successfully done
```

Clear OEM2KEY

Use the following command:

```
> STM32_Programmer_CLI.exe -c port=swd mode=hotplug -lockrdp2 0xFFFFFFFF  0xFFFFFFFF
Device name : STM32U575/STM32U585
Flash size  : 2 MBytes
Device type : MCU
Device CPU  : Cortex-M33
BL Version  : 0x30
Debug in Low Power mode enabled
Lock RDP2 password successfully done
```

The user can check that the OEM1LOCK and OEM2LOCK bits are cleared by reading the content of the FLASH_NSSR register as in Step3 (Section 10.3).

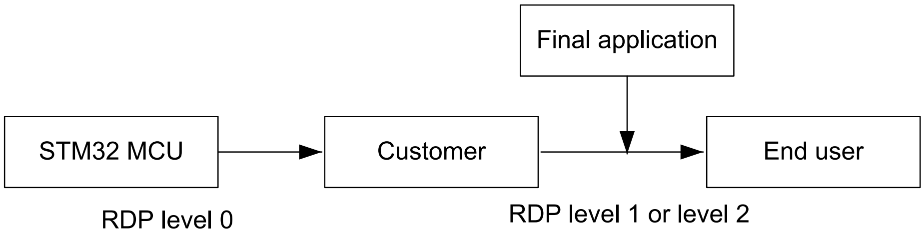
11 Development recommendations using TrustZone

11.1 Development approaches

There are two developer approaches:

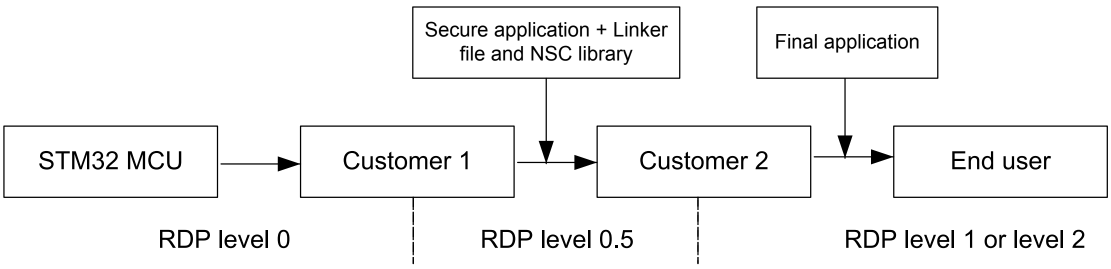
- Single-developer approach: The developer (customer) is in charge of developing secure and nonsecure application. The user application can be protected by using RDP level 1 or RDP level 2.

Figure 12. **Single-developer approach**



- Dual-developer approach: The first developer (customer 1) is in charge to develop the secure application and its associated nonsecure callable library (`.lib`/`.h`), and to provide a predefined linker file to the second developer (customer 2) who is in charge to develop the nonsecure application. This secure application is then loaded in the STM32L5/U5/U3 secure flash memory and protected using RDP level 0.5 to prevent further access to the secure memory region of the device. The second developer (customer 2) then starts his development on a preprogrammed STM32L5/U5/U3 using a linker file and the nonsecure callable library provided by customer 1.
  When the RDP level is set to level 0.5, the customer 1 who provisions the secure flash memory part, must also think about the way to enable the JTAG/SWD for the nonsecure side (for customer 2) after booting in the secure side. For this reason, the customer 1 must implement a switch function that handovers to the nonsecure flash memory, to allow the customer 2 to develop the nonsecure part, and then may be to lock the device to RDP level 1 or level 2.

Figure 13. **Dual-developer approach**



For more details, refer to sections 'Product life-cycle' and 'Software intellectual property protection and collaborative development' in the document [1] or [2].

11.2 Using nonsecure peripherals

When a peripheral is allocated to the nonsecure world, both secure and nonsecure applications can access the peripheral registers.

In the nonsecure world , TrustZone for Armv8-M considerations are totally transparent for the developer. On the secure-world side, the application must ensure that all system resources required by the peripheral are preconfigured, or available to the nonsecure world (such as GPIO, NVIC or DMA).

11.3 Using secure peripherals

When a peripheral is allocated to the secure world, only secure register accesses are granted. Interrupt handling must be managed in the secure world only. Two different software development approaches can be adopted depending on the software interaction requirements between secure and nonsecure projects to use this peripheral.

When working with peripherals that do not require specific interaction with the nonsecure world, the secure world drives them as a standard peripheral without any specific considerations.

If interactions between nonsecure and secure worlds are required to drive the secure peripherals, the secure application must provide nonsecure callable APIs and callbacks to the nonsecure world.

12 Conclusion

The Arm TrustZone technology partitions hardware into secure and nonsecure worlds.

Through the IDAU that defines fixed memory-map security attribution with the user configurable SAU and other features (in flash memory and GTZC), all of STM32 MCU resources are configurable in both secure and nonsecure worlds, including the memory map, the flash memory, SRAM, external memories, peripherals, and peripheral interrupts.

Revision history

Table 13. Document revision history

Date	Version	Changes
11-Oct-2019	1	Initial release.
14-Oct-2019	2	Updated Figure 6. Memory and peripheral data access rules summary .
10-Feb-2020	3	Updated: • Introduction • Section 2.1 Overview • Section 2.2 Security states • Section 3.1 Activation of STM32L5 Series TrustZone • Section 3.3 SAU and IDAU • Figure 3. Address security attribution • Section 3.3.1 STM32L5 Series IDAU and memory aliasing • Table 1. IDAU memory map address security attribution on STM32L5 Series • Section 3.3.2 STM32L5 Series SAU • Section 4 Security configurations on STM32L5 Series devices • Section 4.1 Security configuration of the Flash memory • Section 4.1.1 Secure watermark of the Flash memory • Section 4.1.2 Flash memory block based • Section 5.1 Default security status • Section 5.2 Memory and peripheral security access rules • Figure 6. Memory and peripheral data access rules summary • Section 6 Boot and root secure services (RSS) • Section 7.1 RDP level 1 and Section 7.2 RDP level 0.5 • Section 8 Security features available only when TrustZone is enabled • Section 9 TrustZone deactivation • Section 10.1, Section 10.2 and Section 10.3 • Section 11 Conclusion Added: • Section 1.1 Reference documents • Figure 5. Default Flash bank security state though the option bytes • Table 4. Instruction fetch rules • Section 10 Development recommendations using TrustZone
2-Mar-2020	4	Updated: • Section 7.1 RDP level 1 • Section 7.2 RDP level 0.5 Added Section 9.1 TrustZone/RDP deactivation demonstration using the STM32CubeProgrammer.
28-Sep-2021	5	Updated: • Title and whole text to integrate the STM32U5 Series • Reference documents in Section 1 General information • Section 3.3.2 STM32L5 and STM32U5 SAU • Section 4 Security configurations on STM32L5 and STM32U5 Series • Section 7 Readout protection (RDP) when TrustZone enabled • Section 8 Security features available only when TrustZone is enabled • Section 9.1.2 TZEN/RDP regression with a boot from RSS • Section 11.1 Development approaches Added: • Section 7.3 RDP level 2 • Section 7.4 RDP transitions with OEM keys (STM32U5 only) • Section 9.1.3 TZEN/RDP regression when RDP level 1 is locked by an OEM1 Key • Section 10 Demonstration of RDP transitions using OEM keys (STM32U5 only)
08-Apr-2022	6	Updated: • Various typo

Date	Version	Changes
24-Feb-2025	7	• Updated the entire document to introduce STM32U3 series. • Updated the document title. • Corrected various typos.

Contents

List of tables

List of figures