
STM32L5 Series TrustZone® features

Introduction

In the IoT (internet of things) application, 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.

The STM32L5 Series devices (named STM32L5 later in this document) are based on the high-performance Arm® Cortex®-M33 32-bit RISC core. The processor is based on the Armv8-M architecture and is primarily for use in environments where security is an important consideration.

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

This document introduces the TrustZone® technology and the features of STM32L5 Series devices that allow the partition of STM32L5 Series memory/resources between secure and non-secure.

1 General information

This application note applies to the STM32L5 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

The following documents must be referred to while reading this application note:

- [1] Reference manual *STM32L552xx and STM32L562xx advanced Arm®-based 32-bit MCUs* (RM0438)
- [2] Armv8-M Architecture Reference Manual available from the Arm® web site.

2 TrustZone technology

2.1 Overview

The Arm TrustZone technology for Aemv8-M partitions the system into two regions: one is secure world and another is non-secure world.

The division of secure and non-secure 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 non-secure world. After planning the security attribution of these resources, non-secure world only accesses non-secure memories and resources, while secure world is able to access all memories and resources in both worlds, including secure and non-secure resources.

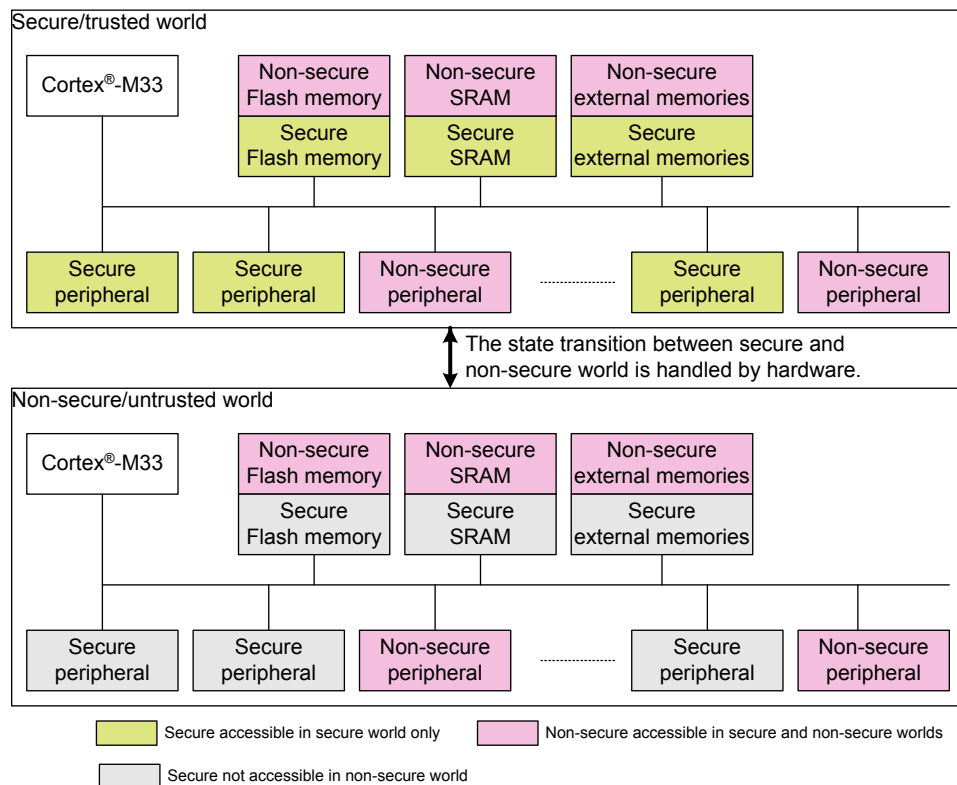
Important data that needs protection (such as cryptographic keys), must be placed and processed in the secure world safely.

The location the code is executed in defines its type:

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

Secure and non-secure codes run on the same STM32L5 device, as illustrated in the figure below.

Figure 1. Resources partition between secure and non-secure 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 non-secure:

- If the CPU runs code in a non-secure memory, the CPU is in non-secure 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.
- **Non-secure 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. To prevent non-secure applications from branching into invalid entry points, there is the SG instruction.

When a non-secure 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 non-secure callable functions to provide secure service accesses to non-secure code.

- **Non-secure**
Non-secure addresses are used for memory and peripherals accessible by all software running on the device. Non-secure transactions are those that originate from masters operating as non-secure or from secure masters accessing a non-secure address. Non-secure transactions are only permitted to access non-secure addresses, the non-secure transactions are denied access to secure addresses.

3 TrustZone implementation on STM32L5 Series

3.1 Activation of STM32L5 TrustZone

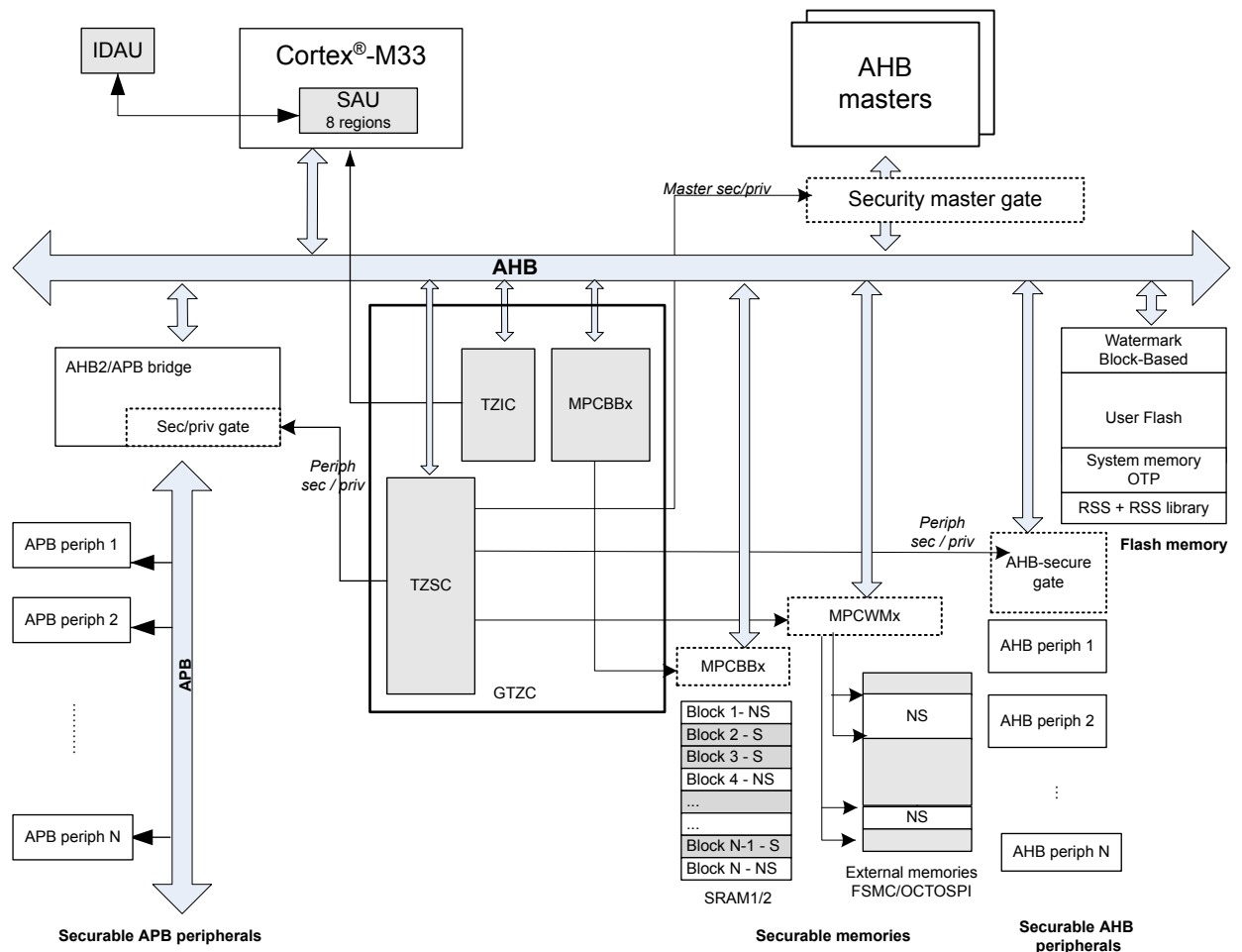
In the STM32L5, the TrustZone is disabled by default and enabled by setting the TZEN option bit in the corresponding option byte.

All the features described in this section and the rest of the document are applied to the STM32L5 devices with TrustZone enabled.

3.2 TrustZone block diagram

In the STM32L5, 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 TrustZone implementation overview



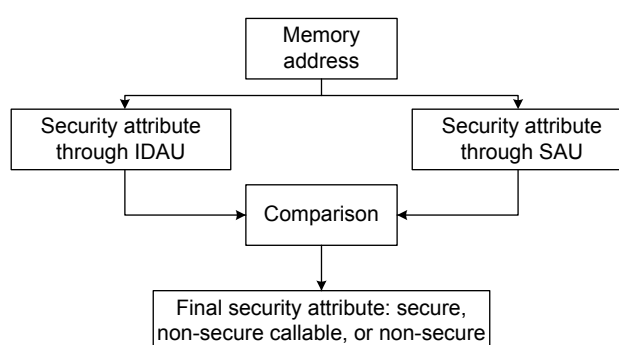
3.3 Secure attribution unit (SAU) and implementation defined attribution unit (IDAU)

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.
- Non-secure callable has lower secure priority.
- Non-secure has the lowest secure priority.

Figure 3. Address security attribution



3.3.1 STM32L5 IDAU and memory aliasing

The STM32L5 memory mapping follows the Arm recommendations to implement a duplicated memory map, one for the secure view and the other for the non-secure view.

This means that each region of the memory map (code, SRAM, peripherals) is divided into two sub-regions where internal memories and peripherals are decoded at two separate address locations, in the non-secure 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 IDAU.

Table 1. IDAU memory map address security attribution on STM32L5 Series

Region	Address range	Security attribute through IDAU
Code-external memories when remapped	0x0000 0000-0x07FF FFFF	Non-secure
Code-Flash memory and SRAM	0x0800 0000-0x0BFF FFFF	Non-secure
	0x0C00 0000-0x0FFF FFFF	Non-secure callable
Code-external memories when remapped	0x1000 0000-0x1FFF FFFF	Non-secure
SRAM	0x2000 0000-0x2FFF FFFF	Non-secure
	0x3000 0000-0x3FFF FFFF	Non-secure callable
Peripherals	0x4000 0000-0x4FFF FFFF	Non-secure
	0x5000 0000-0x5FFF FFFF	Non-secure callable
External memory ⁽¹⁾	0x6000 0000-0xDFFF FFFF	Non-secure

1. The external memory area is not aliased.

3.3.2

STM32L5 Series SAU

There are eight SAU regions in the STM32L5. The user changes the required security configuration partition by SAU as shown in the table below. When the SAU is disabled, all the memory regions are considered as secure.

Table 2. SAU memory map address security attribution on STM32L5 Series

Region	Address range	Security attribute through IDAU	Security attribute through SAU	Final security attribute
Code-external memories when remapped	0x0000 0000 - 0x07FF FFFF	Non-secure	Secure Non-secure or Non-secure callable	Secure Non-secure or Non-secure callable
Code-Flash memory and SRAM	0x0800 0000 - 0x00BFF FFFF	Non-secure	Non-secure	Non-secure
	0x0C00 0000 - 0x0FFF FFFF	Non-secure callable	Secure or Non-secure callable	Secure or Non-secure callable
Code-external memories when remapped	0x1000 0000 - 0x1FFF FFFF	Non-secure	Non-secure	Non-secure
SRAM	0x2000 0000 - 0x2FFF FFFF	Non-secure	Non-secure	Non-secure
	0x3000 0000 - 0x3FFF FFFF	Non-secure callable	Secure or Non-secure callable	Secure or Non-secure callable
Peripherals	0x4000 0000 - 0x4FFF FFFF	Non-secure	Non-secure	Non-secure
	0x5000 0000 - 0x5FFF FFFF	Non-secure callable	Secure or Non-secure callable	Secure or Non-secure callable
External memories	0x6000 0000 - 0xDFFF FFFF	Non-secure	Secure Non-secure or Non-secure callable	Secure Non-secure or Non-secure callable

For instance, a peripheral is decoded at two address ranges: in the 0x4000 0000 non-secure view and in the 0x5000 0000 secure view.

Then, 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 non-secure code accesses the same peripheral at another address in the non-secure view. Then, 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

The definition of the SAU regions is made in the CMSIS files `Device partition_stm32l552xx.h` and `partition_stm32l562xx.h`.

The secure project enables the SAU and defines the SAU regions.

The STM32CubeL5 defines the following default SAU regions associated with linker memory layout file templates:

- SAU region 0: 0x0C03 E000-0x0C03 FFFF (secure, non-secure callable)
- SAU region 1: 0x0804 0000-0x0807 FFFF (non-secure 256-Kbyte Flash Bank2)
- SAU region 2: 0x2001 8000-0x2003 FFFF (non-secure SRAM, 160-Kbyte second-half SRAM1 + SRAM2)
- SAU region 3: 0x4000 0000-0x4FFF FFFF (non-secure peripherals mapped memory)
- SAU region 4: 0x6000 0000-0x9FFF FFFF (non-secure external memories)
- SAU region 5: 0x0BF9 0000-0x0BFA 8FFF (non-secure system memory)
- SAU region 6: not used
- SAU region 7: not used

All memory space in 0x0000 0000-0xDFFF FFFF not covered by an SAU region, is by default 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 table below.

Table 3. STM32CubeL5 memory security partitioning

Region	Address range	Security attribute through IDAU	Security attribute through SAU	Final security attribute
Flash memory	0x0804 0000 - 0x0807 FFFF	Non-secure	Non-secure	Non-secure
	0x0C00 0000 - 0x0C03 DFFF	Non-secure callable	Secure	Secure
	0x0C03 E000 - 0x0C03 FFFF	Non-secure callable	Non-secure callable	Non-secure callable
SRAM1	0x3000 0000 - 0x3001 7FFF	Non-secure callable	Secure	Secure
	0x2001 8000 - 0x2002 FFFF	Non-secure	Non-secure	Non-secure
SRAM2	0x2003 0000 - 0x2003 FFFF	Non-secure	Non-secure	Non-secure
Peripherals	0x4000 0000 - 0x4FFF FFFF	Non-secure	Non-secure	Non-secure
	0x5000 0000 - 0x5FFF FFFF	Non-secure callable	Secure	Secure
External memories	0x6000 0000 - 0x9FFF FFFF	Non-secure	Non-secure	Non-secure

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

4 Security configurations on STM32L5 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 devices come with complementary security features that reinforce and allow a more flexible partition between the secure and the non-secure 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 non-volatile Flash secure watermark and volatile block based Flash interface registers, even if they are non-secure through IDAU/SAU.

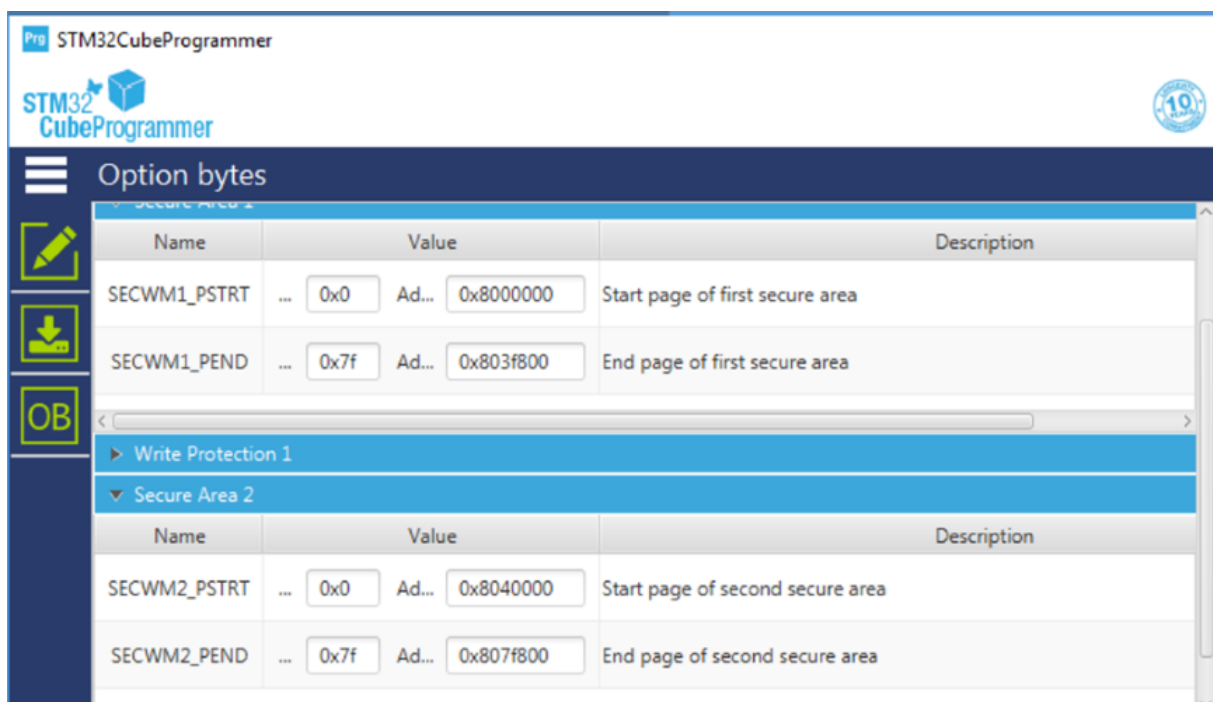
SAU and IDAU are responsible to grant the transactions issued by the CPU. The Flash secure watermarks and block-based grant the transactions from the STM32L5 CPU and other masters (such as DMA1, DMA2 or SDMMC). As seen in Figure 2, each transaction, issued by Cortex-M33 that targets Flash memory, is first checked by IDAU/SAU then checked by Flash secure watermark or block based registers. Refer to Figure 6 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: SECWMx_PSTRT and SECWMx_PEND (x = 1, 2).

The default value after setting TZEN, corresponds to all the Flash memory is secure.

Figure 4. Default Flash memory state through the option bytes



The STM32CubeL5 TrustZone examples are assuming that Bank1 is secure and Bank2 is non-secure.

Figure 5. Default Flash bank security state through the option bytes

Name	Value	Address	Description
SECWM1_PSTRT	Value: 0x0	Address: 0x8000000	Start page of first secure area
SECWM1_PEND	Value: 0x7f	Address: 0x803f800	End page of first secure area
Write Protection 1			
Secure Area 2			
Name	Value	Address	Description
SECWM2_PSTRT	Value: 0x1	Address: 0x8040800	Start page of second secure area
SECWM2_PEND	Value: 0x0	Address: 0x8040000	End page of second secure area

4.1.2 Flash memory block based

Even if all the Flash memory is non-secure through IDAU/SAU and through the Flash secure watermark option bytes, it is possible to configure volatile secure areas using the Flash memory block based feature.

Any page is programmed on the fly either as secure or non-secure mode, using the Flash interface block based configuration registers.

Block based registers can only set a page as secure whereas it is set as non-secure through Flash secure watermark option bytes. The contrary is not possible, this means that it is not possible to configure using block based registers a page as non-secure when it is configured as secure through Flash secure watermark option bytes.

4.2 Global TrustZone controller (GTZC)

The GTZC has the following sub-blocks:

- TZSC (TrustZone security controller) allows the security attribute configuration of:
 - Peripherals (see the note below) as either secure or non-secure.
 - External memories region: through watermark memory protection controller (watermark memory protection controller MPCWMx, x = 1, 2, 3)
- MPCBB (block-based memory protection controller) allows the security attribute configuration of the SRAM1 and SRAM2 blocks.
- 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 set to either securable or TrustZone-aware:

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

For the list of securable and TrustZone-aware peripherals, refer to the document [1].

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 be in a secure address.
- 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 non-secure resources..
- The Flash memory security area is defined by watermark user option bytes, for which production values are:
 - SECWMx_PSTRT = 0x00
 - SECWMx_PEND = 0x7F (x = 1,2).

This means that all Flash memory is secure.

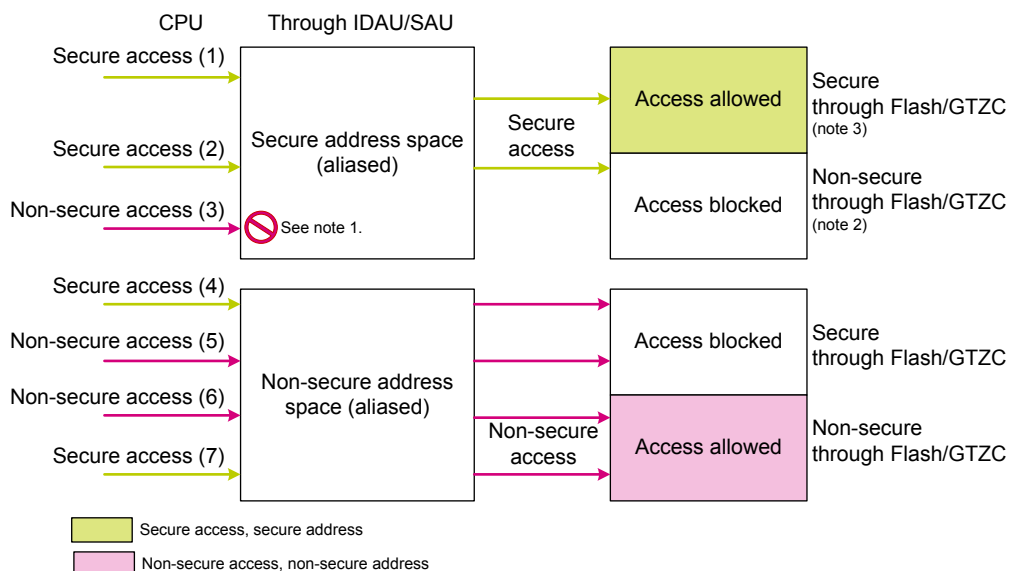
- All SRAMs are secure.
- External memories: FSMC and OCTOSPI banks are secure.
- All peripherals (except GPIOs) are **non-secure**.
- All GPIOs are secure.
- All DMA channels are **non-secure**.
- Backup registers are **non-secure**.

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 interface, SRAM, external memory or any secure peripheral) .

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

Figure 6. Memory and peripheral data access rules summary



- Notes:
1. An access blocked by SAU/IDAU, results in secure fault.
 2. Non-secure registers of peripherals are accessible by the secure transactions as well. This is a difference with the memory access rules. Also, when the SRWLADIS bit in GTZC_MPCBBx_CR register is set, secure read/write access is allowed on non-secure SRAM block.
 3. Secure / non-secure through Flash / GTZC refers to security attribute through Flash secure watermark option Bytes, Flash memory block based registers, MPCWMx, MPCBB 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 the SRAMs, external memories and peripherals).

Note:

The non-secure information block is only accessible by non-secure transactions.

In particular, the OTP area, VREFINT and temperature sensor calibration values are only accessed by non-secure transactions. So, the secure application must program an SAU region configuring this area as non-secure.

The access rules are the following:

- In case of a secure access to an address that is secure through SAU/IDAU and secure through Flash/GTZC, the access is allowed. See (1) in the above figure.
- In case of a secure access to an address that is secure through SAU/IDAU and non-secure through Flash/GTZC, the access is blocked. See (2) in the above figure.
- In case of a non-secure 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 the above figure.
- In case of a secure access to an address that is non-secure through SAU/IDAU and secure through Flash/GTZC, the access is blocked. See (4) in the above figure.
- In case of a non-secure access to an address that is non-secure through SAU/IDAU and secure through Flash/GTZC, the access is blocked. See (5) in the above figure.
- In case of a non-secure access to an address that is non-secure through SAU/IDAU and non-secure through Flash/GTZC, the access is allowed. See (6) in the above figure.
- In case of a secure access to an address that is non-secure through SAU/IDAU and non-secure through Flash/GTZC, the access is allowed. See (7) in the above figure.

When the access is blocked, the result is either:

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

For example, a non-secure 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 detailed information, refer to the document [1].

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

Table 4. Instruction fetch rules

CPU state	Target memory address security attribute through IDAU/SAU	Transaction
Secure or non-secure	Non-secure	Non-secure
Secure or non-secure	Secure or non-secure callable	Secure
Secure or non-secure	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].

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 option bytes. 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.

Table 5. 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. SECBOOTADD0[24:0] option bytes are used to select the boot secure memory address. To increase the security and establish a root of trust, 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.

Caution: Once set, the BOOT_LOCK option bit cannot be cleared. That means that the unique boot entry address is the address programmed in SECBOOTADD0[24:0] option bytes.

7 Readout protection (RDP)

There are four levels of readout protection from no protection (level 0) to maximum protection or no debug (level 2), detailed in the table below.

Table 6. RDP protection levels summary

RDP byte value	RDP byte complement value	RDP level
0xAA	0x55	0
0x55	0xAA	0.5
Any value except 0x55, 0xAA, or 0xCC	Any value (not necessary complementary) except 0x55, 0xAA, or 0xCC	1
0xCC	0x33	2

7.1 RDP level 1

In RDP level 1, the Flash main memory, the backup registers and the SRAM2 are totally inaccessible. An intrusion is detected in case of debug access when the CPU is in secure state.

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

The RDP regression must be done through bootloader or JTAG / SWD, as detailed below:

- 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, with a boot from the user Flash memory: In order to be able to connect to the target, the CPU must be in non-secure state.

Before programming the RDP level 1, the user must ensure that the secure application calls the non-secure application so that the connection to the target is possible, and the regression from RDP level 1 is possible.

Caution: If there is no non-secure 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/bootloader with a boot from RSS. For more details, refer to the Boot configuration section of the document [1].

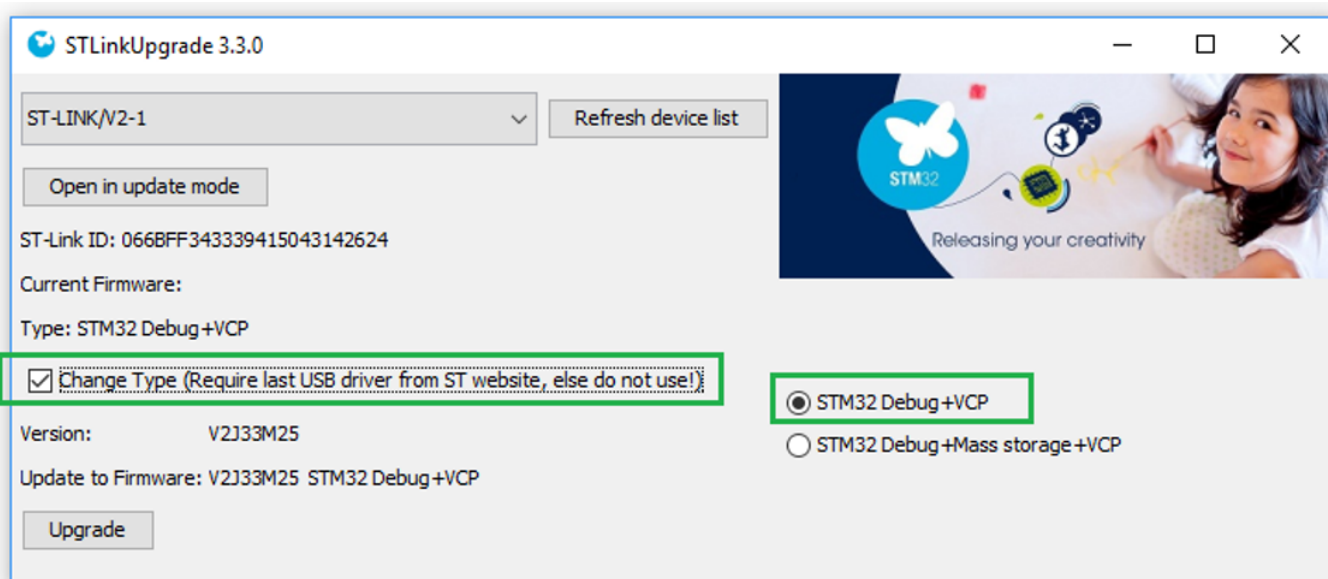
The ST boards for STM32L5 Series (Nucleo, Evaluation and Discovery) come with the integrated ST-LINK that can be used as a power supply and for debug at the same time. Due to the mass storage interface of the ST-LINK that requires the target identification at ST-LINK startup (thus SWD connection), a debug intrusion is detected every time the ST-LINK USB cable is plugged. Consequently, the user application is never executed and the CPU remains always in secure state. 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:

- 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 STLinkUpgrade application, as shown in the figure below.

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

Figure 7. 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 and SRAMs area are inaccessible. The non-secure Flash memory, the non-secure backup registers and the non-secure SRAMs area remain accessible. When the CPU is in secure state, it is not possible to connect to the target through JTAG.

When the CPU is in non-secure state, connection to the target through JTAG and RDP regression are possible.

The RDP regression is done through either bootloader or JTAG, as detailed below:

- 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, with a boot from the user Flash memory: In order to be able to connect to the target, the CPU must be in non-secure 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 non-secure state.

Before programming the RDP level 0.5, the user must always ensure that the secure application calls the non-secure application so that the connection to the target is possible, and the regression from RDP level 0.5 is possible.

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

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].

Note: In RDP levels 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.
 - SECBOOTADD0[24:0] is an address in the secure user Flash memory.
 - There is no non-secure code, so the CPU is always in secure state and it is not possible to program the non-secure Flash memory in RDP level 0.5.

8 Security features available only when TrustZone is enabled

The following features are available only when TrustZone is enabled:

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

9 TrustZone deactivation

As mentioned in [Section 2.1 Overview](#), the TrustZone is disabled by default in all STM32L5 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 (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: *If the BOOT_LOCK option bit is set, it cannot be cleared. This means that 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.*

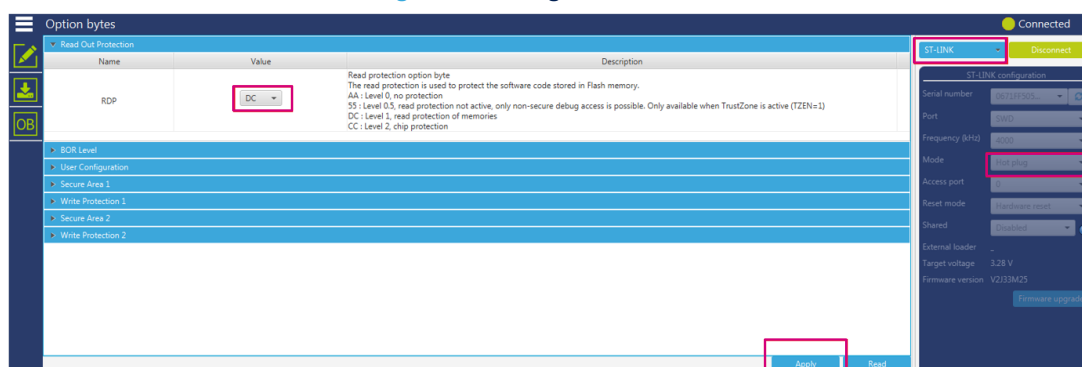
9.1 TrustZone/RDP deactivation demonstration using the STM32CubeProgrammer

9.1.1 TZEN/RDP regression with a boot from user Flash

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

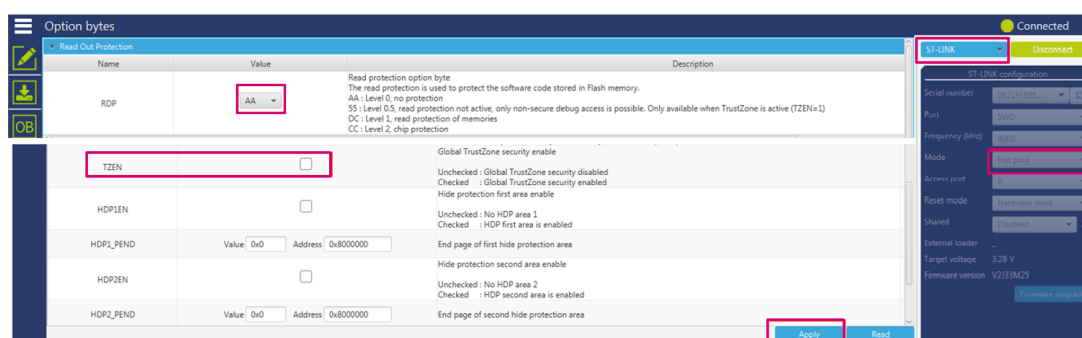
1. Make sure that secure and non-secure applications are well loaded and executed.
2. Set RDP to level 1 (or level 0.5). Then only Hotplug connection is possible.

Figure 8. Setting RDP to level 1



3. If RDP is set to level 1, use a power supply different from ST-LINK (more details in [Section 7.1 RDP level 1](#)) in order to be able to connect to the target (not needed with RDP level 0.5).
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 non-secure 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

How changing the boot on STM32L5 ST boards?

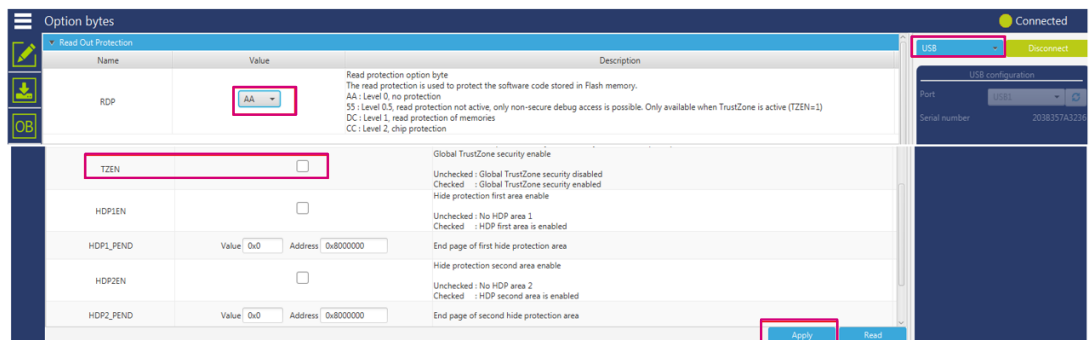
The boot must be done from RSS by applying a high level on BOOT0 pin:

- On STM32L5 evaluation board (STM32L552E-EV), a switch SW1 is provided to change the boot. Refer to the user manual *Evaluation board with STM32L552ZE MCU* (UM2597).
- On STM32L5 Discovery kit (STM32L562E-DK), a small rework must be done to change the boot from RSS. Refer to the user manual *Discovery kit with STM32L562QE MCU* (UM2617).
- On STM32L5 Nucleo board (NUCLEO-L552ZE-Q), a connection between CN11 pin 5 (VDD) and pin 7 (PH3_BOOT0) must be done. Refer to the user manual *STM32L5 Nucleo-144 board* (UM2581).

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

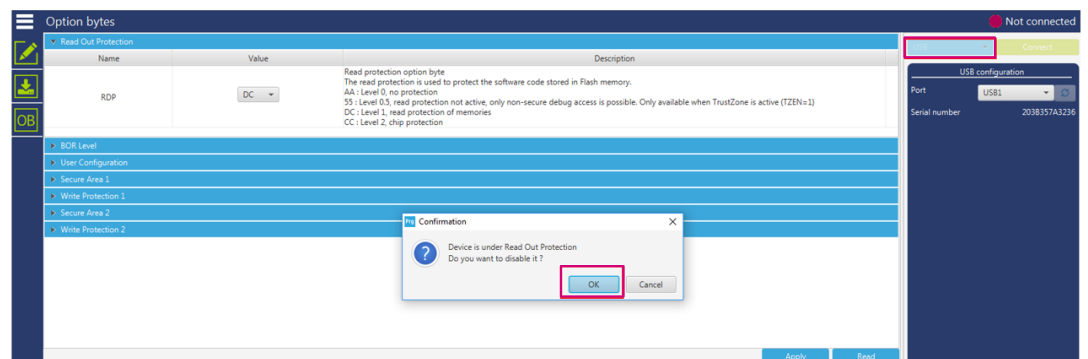
- Through JTAG/SWD, set RDP to level 0 (option byte value 0xAA) and uncheck the *TZEN* box, then click on *Apply*.
- Through bootloader, using one of the supported communication interfaces (USB in this example):
 - If RDP is set to level 0.5, set RDP level to level 0 (option byte value 0xAA) and uncheck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

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



- If RDP is set to level 1, it is only possible to do RDP regression with the STM32CubeProgrammer graphical interface.

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



If RDP is set to level 1, TZEN regression cannot be done using the STM32CubeProgrammer graphical interface. In this case, STM32CubeProgrammer CLI must be used, applying the following TZEN regression command:

Figure 12. TZEN regression using STM32CubeProgrammer CLI

```
C:\Program Files (x86)\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin>STM32_Programmer_CLI.exe -c port=USB1 -tzenreg
-----
STM32CubeProgrammer v2.3.0
-----

USB speed : Full Speed (12MBit/s)
Manuf. ID : STMicroelectronics
Product ID : DFU in FS Mode
SN : 2038357A3236
FW version : 0x011a
Device ID : 0x0472
Warning: Device is under Read Out Protection

Disabling TrustZone...
Disabling TrustZone successfully
```

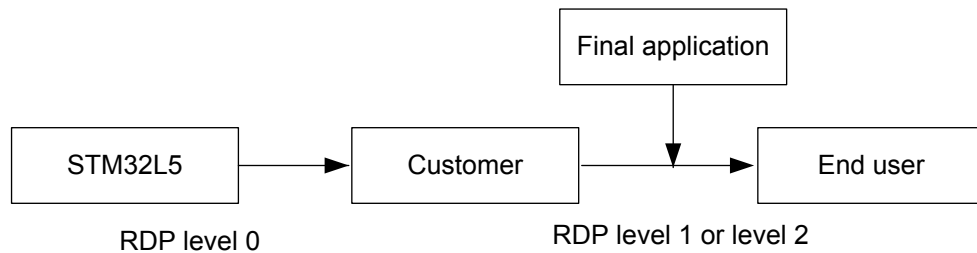
10 Development recommendations using TrustZone

10.1 Development approaches

There are two developer approaches:

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

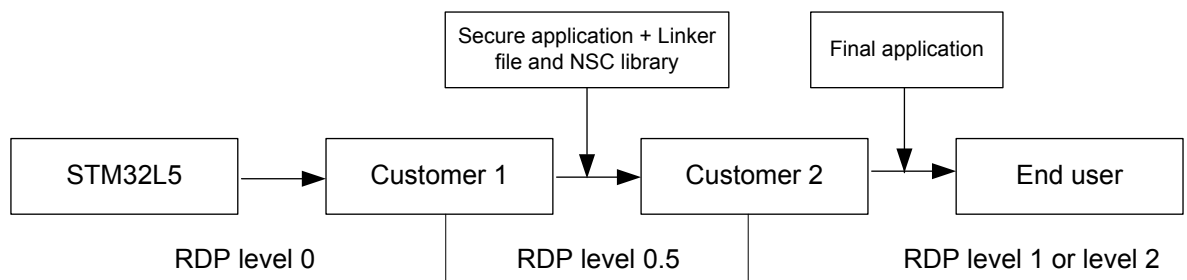
Figure 13. Single-developer approach



- Dual developer approach: The first developer (customer 1) is in charge of developing the secure application and its associated non-secure callable library (.lib/.h), and providing a predefined linker file to the second developer (customer 2) who is in charge of developing the non-secure application. This secure application is then loaded in the STM32L5 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 using a linker file and the non-secure callable library provided by customer 1.

When the RDP level is set to level 0.5, the customer 1 who provisions the secure flash part, must also think about the way to enable the JTAG for the non-secure side (for customer 2) after booting in the secure side. That is why, the customer 1 must implement a switch function which handover to the non-secure flash, to allow the customer 2 developing the non-secure part and then may be to lock the device to RDP level 1 or level 2.

Figure 14. Dual-developer approach



For more details, refer to sections 'Product lifecycle' and 'Software IP protection and collaborative development' in the document [1].

10.2 Using non-secure peripherals

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

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

10.3 Using secure peripherals

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

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

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

11 Conclusion

The Arm TrustZone technology partitions hardware into secure and non-secure 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 STM32L5 Series resources are configurable in both secure and non-secure worlds, including the memory map, the Flash memory, SRAM, external memories, peripherals and peripheral interrupts.

Revision history

Table 7. 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	<p>Updated:</p> <ul style="list-style-type: none"> • 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 <p>Added:</p> <ul style="list-style-type: none"> • 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	<p>Updated:</p> <ul style="list-style-type: none"> • Section 7.1 RDP level 1 • Section 7.2 RDP level 0.5 <p>Added Section 9.1 TrustZone/RDP deactivation demonstration using the STM32CubeProgrammer.</p>

Contents

1	General information	2
2	TrustZone technology	3
2.1	Overview	3
2.2	Security states	4
3	TrustZone implementation on STM32L5 Series	5
3.1	Activation of STM32L5 TrustZone	5
3.2	TrustZone block diagram	5
3.3	Secure attribution unit (SAU) and implementation defined attribution unit (IDAU)	6
3.3.1	STM32L5 IDAU and memory aliasing	6
3.3.2	STM32L5 Series SAU	7
4	Security configurations on STM32L5 Series	9
4.1	Security configuration of the Flash memory	9
4.1.1	Secure watermark of the Flash memory	9
4.1.2	Flash memory block based	10
4.2	Global TrustZone controller (GTZC)	10
5	Overall system security access rules	11
5.1	Default security status	11
5.2	Memory and peripheral security access rules	11
6	Boot and root secure services (RSS)	13
7	Readout protection (RDP)	14
7.1	RDP level 1	14
7.2	RDP level 0.5	15
8	Security features available only when TrustZone is enabled	17
9	TrustZone deactivation	18
9.1	TrustZone/RDP deactivation demonstration using the STM32CubeProgrammer	18
9.1.1	TZEN/RDP regression with a boot from user Flash	18
9.1.2	TZEN/RDP regression with a boot from RSS	19
10	Development recommendations using TrustZone	21
10.1	Development approaches	21

10.2	Using non-secure peripherals	22
10.3	Using secure peripherals	22
11	Conclusion	23
	Revision history	24
	Contents	25
	List of tables	27
	List of figures	28

List of tables

Table 1.	IDAU memory map address security attribution on STM32L5 Series	6
Table 2.	SAU memory map address security attribution on STM32L5 Series	7
Table 3.	STM32CubeL5 memory security partitioning	8
Table 4.	Instruction fetch rules	12
Table 5.	Boot space versus RDP protection	13
Table 6.	RDP protection levels summary	14
Table 7.	Document revision history	24

List of figures

Figure 1.	Resources partition between secure and non-secure worlds	3
Figure 2.	STM32L5 TrustZone implementation overview	5
Figure 3.	Address security attribution	6
Figure 4.	Default Flash memory state though the option bytes	9
Figure 5.	Default Flash bank security state though the option bytes.	10
Figure 6.	Memory and peripheral data access rules summary	11
Figure 7.	Disabling the mass storage interface of the ST-LINK	15
Figure 8.	Setting RDP to level 1	18
Figure 9.	TZEN and RDP regression through SWD with boot from user Flash	18
Figure 10.	TZEN and RDP regression (level 0.5 to level 0) through bootloader.	19
Figure 11.	RDP regression (level 1 to level 0) through bootloader.	19
Figure 12.	TZEN regression using STM32CubeProgrammer CLI	20
Figure 13.	Single-developer approach.	21
Figure 14.	Dual-developer approach.	21

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2020 STMicroelectronics – All rights reserved