HDP secure area for STM32H7B3xx microcontrollers

Introduction

This application note describes the hardware-protection feature (HDP) for the STM32H7B3xx microcontrollers, and includes guidelines for the STM32CubeH7 example code, that runs on the STM32H7B3I-EVAL evaluation board.

Table 1. **Applicable products**

Type	Products
STM32H7B3xx	STM32H7B3AI, STM32H7B3II, STM32H7B3LI, STM32H7B3NI, STM32H7B3QI, STM32H7B3RI, STM32H7B3VI, STM32H7B3ZI

1 Secure HDP overview

This section introduces the secure-area concept in STM32H7B3xx Arm® Cortex®- based devices, to be understood before trying out the tutorial detailed in Section 2 Secure area, STM32CubeH7 example for STM32H7B3xx.

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

1.1 Secure user-memory area

A secure user-memory area (also called secure HDP), is a Flash memory area defined by the user, that is executed only once just after the boot.

A Flash memory area, when protected, cannot be read anymore by a user.

Two secure areas, one in each Flash bank, can be defined for the STM32H7B3xx devices.

The user can include his own code in these Flash memory areas before protecting them. When the Flash memory area is protected, the code is executed in a secure way, and only the CPU accesses to the defined Flash areas. No direct data transfers from these protected Flash areas are possible. If the debugger is not specifically activated by the secure code, the debugger access is possible only after execution of the protected code.

Typical applications are code signature or integrity checking (user secure boot), software license checking, secure firmware update.

A call to a specific RSS service is used to jump out of a secure area and to start the execution of the code stored in the conventional user-memory area.

The code stored in the secure user-memory area is hidden after the jump to the non-secure application code.

If the code in the secure area activates the debugger, this code can be read and modified even if the secure area has been set. This can be useful during debugging.

If the debugger is only activated when jumping out of the secure area, the secure code cannot be read or modified anymore. A modification then requires to first erase the Flash memory through a regression and to load the new code.

A regression can be done through a debugger or by the code located in the secure or non-secure area.

It is important that the secure user code is executed correctly and extensively tested before setting a secure area. If no debugger is activated and no regression can be done, the device can be definitively locked.

Note: For all STM32H7xxxx devices, the debugger can be activated or not (argument of exitSecureArea RSS service) after the secure code execution, except for the STM32H75xxx MCUs, for which the debugger is automatically activated after the secure code execution.

Section 2 provides an example with all the advised intermediate steps. It is stronly recommended to follow all these steps, even for the user code development.

1.2 Secure access mode (security option bit)

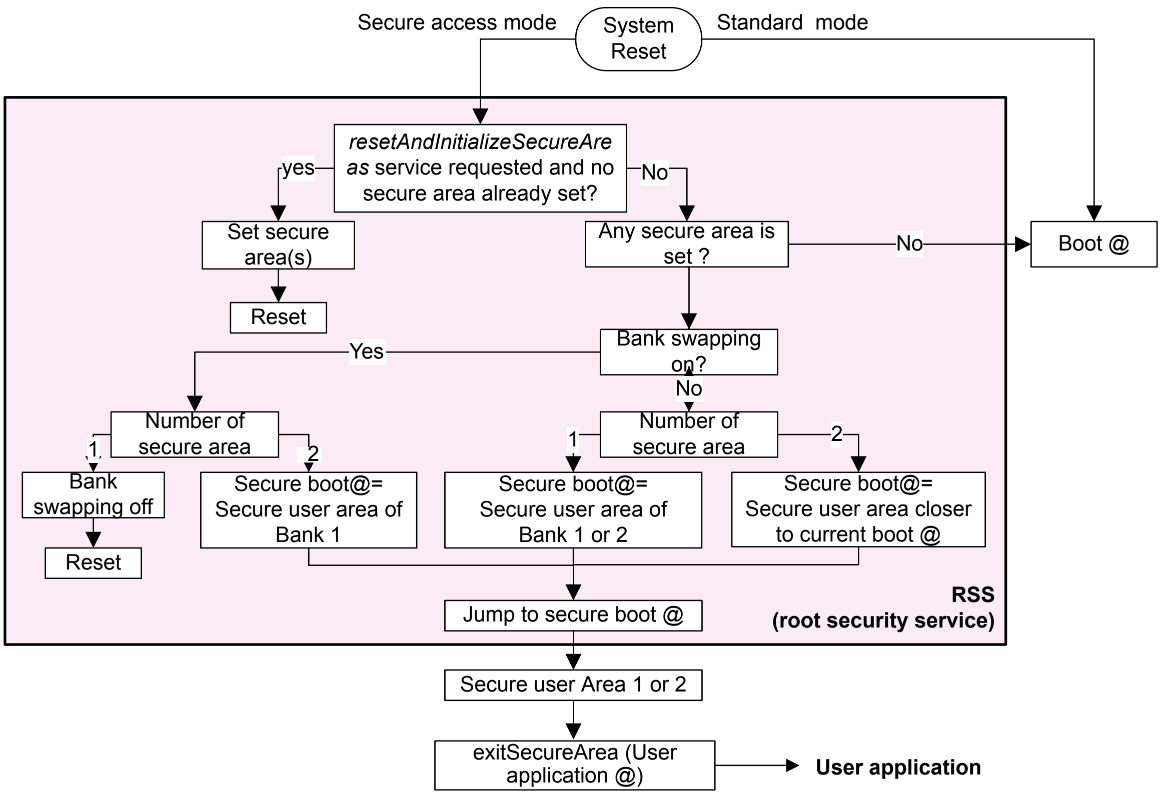
Before enabling a secure area, the device must operate in secure access mode (read the FLASH section of the reference manual for more details). By default, a device does not operate in secure access mode. To enable this mode, the security option bit needs to be set.

Figure 1 gives a summary of the possible boot ways, and the impact of setting the security option bit. If the security bit is not set, the boot is done through the standard boot mode: jump to the boot address that is configured through the BOOT pin and the BOOT_ADD0/1 option bytes.

In secure access mode, the device boots always at a unique boot entry point in the secure bootloader. If no secure services are required, the standard boot is executed.

The figure below illustrates also the cases where a secure area programming is requested or a secure area is already set.

Figure 1. Bootloader state machine in secure access mode (STM32H7B3xx)


Secure access mode
System Reset
Standard mode
resetAndInitializeSecureAreas service requested and no secure area already set?
yes
No
Set secure area(s)
Any secure area is set ?
No
Boot @
Reset
Bank swapping on?
Yes
No
Number of secure area
1
2
Number of secure area
1
2
Bank swapping off
Secure boot@= Secure user area of Bank 1
Secure boot@= Secure user area of Bank 1 or 2
Secure boot@= Secure user area closer to current boot @
Reset
Jump to secure boot @
RSS (root security service)
Secure user Area 1 or 2
exitSecureArea (User application @)
User application

Note: *The RSS is located in the Flash system memory.*

One of the first step of the tutorial in Section 2 is to set the secure option bit using STM32CubeProgrammer. As shown in Figure 1, as long as no secure area is set or requested, the user does not notice any difference, since the standard boot is executed. But the root security service (RSS) is executed first after the system reset.

Resetting the security option bit can only be done if the secure area and the PCROP (proprietary code readout protection) areas are removed (or not yet set).

In the tutorial of Section 2 , it is the last step done after the RDP regression that removes the complete Flash memory content and the protection.

1.3 Secure user memory area setting (address option bits)

A secure area can only be set if the device operates in secure access mode (security option bit set). As soon as one or two secure areas are set, the path in the RSS is quite different as shown in Figure 1.

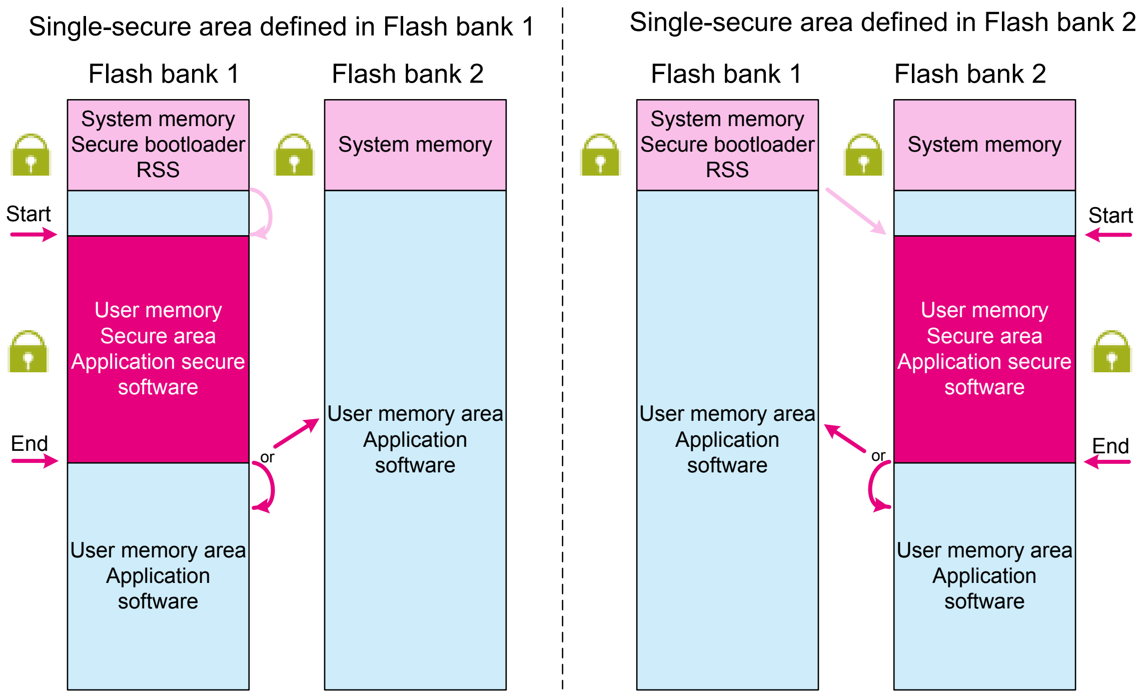
Important:
The developer must define at early state if a bank swap is requested or needed in the future. If only one secure area is defined, the bank swap is not possible. When a secure area is set, only the RSS and a code running in the secure area can modify it or set a second secure area (one secure is possible for each bank). The consequence is that the secure areas can only be set once by STM32CubeProgrammer or a code running in a non-secure area. A modification then requires to first erase the complete Flash memory through an RDP regression.

1.3.1 Case 1: single secure area (no bank swap possible)

If a secure area is set, the RSS jumps to the address of the secure area that can be in the Flash bank 1 or Flash bank 2.

If a bank swap has been set through the option byte programming, the bank swap is removed and a device reset is automatically done. So the device boots again and the RSS is executed once more.

Figure 2. Single secure area



There is no restriction about the secure area start and end address.

To jump to the application code, a specific service of the RSS is called at the end of the secure code execution. This service insures the safe closure of the secure area before jumping to the application code

This jump can be to any address of the user memory bank 1 or bank 2.

The debugger can be activated or not after exiting the secure area.

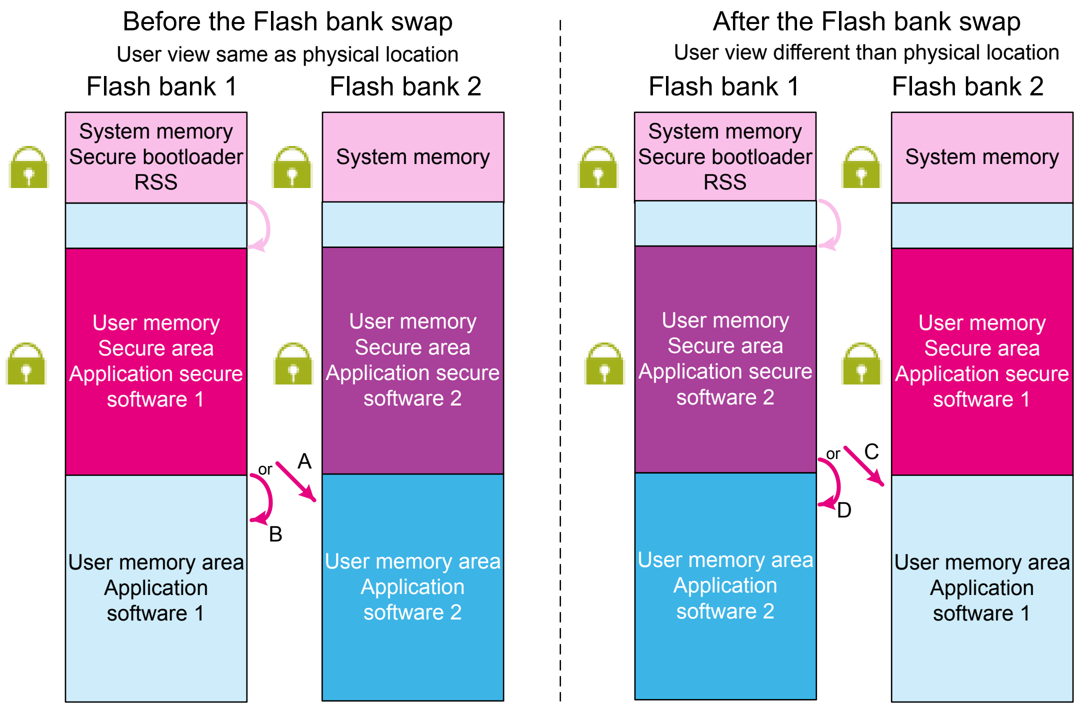
The reference manual (section “Secure area exiting service”) explains this RSS service.

1.3.2 Case 2: two secure areas (one used with bank swap)

The bank swap can be quite confusing for a user. From the user point of view, activating the bank-swap option bit is observed as if the content of the Flash bank 2 is copied in the Flash bank 1, and bank 2 in bank 1. It is the behavior when reading the Flash memory content with, for instance, the STM32CubeProgrammer. But physically the bank 1 and bank 2 content are not moved.

In secure access mode, the secure bootloader located in the Flash bank 1 is always executed first. This is unchanged by the bank swap, as shown in the figure below.

Figure 3. Two secure areas (one used with bank swap)



The above figure shows an example of code execution in Flash bank1.

If an IDE (integrated development environment) is used to flash the user code, the best way is to follow these steps:

- Compile and flash the two software 1 for bank 1.
- Activate the bank swap, by setting the option bit.
- Compile and flash the two software 2 for bank 1.
- Before setting the two secure areas, crosscheck (playing with the bank swap setting) that the softwares are executed as expected, including the jump in the application software.

The jump address out of software 2 must be defined as after the bank swap. So the D step in the above figure must be an address in bank 1 (0x08xxxxxx) and the C step must be in bank 2.

As for a single secure area, if the debugger is not activated and no regression is possible, a failing code in one of the two secure areas can definitively lock the device.

1.4 Secure user memory removal

A secure area removal can only be done through an RDP regression. It is required first to set a global readout protection as explained in Section 2.4.1 .

In a second step, the RDP and the secure area can be removed using the procedure explained in Section 2.4.2 .

Section 2.4 explains the different steps using the STM32CubeProgrammer. The same steps can also be done through a user code.

Important:
The correct polarity of the DMES bit has to be chosen when performing a regression.

If DMES = 0, the content of the secure area is kept. When the device reboots, this remaining secure code is automatically executed. A device can be lost if the jump-out of this secure area is to an erased Flash memory area without other jump-address option.

2 Secure area, STM32CubeH7 example for STM32H7B3xx

This tutorial shows the different steps to set and remove a single secure area for an STM32H7B3xx product. It is available for the STM32H7 Evaluation board and Discovery kit in the STM32CubeMx Applications folder.

Pre-requisite

This section describes the tutorial for the STM32H7B3I-EVAL evaluation board. Refer to the user manual *Evaluation board with STM32H7B3LI MCU* (UM2662) for the detailed description on how to set and use this board.

One of the supported tool chain is needed to compile the code. The STM32CubeProgrammer is used to modify the option bytes.

2.1 Step 1: initial setting

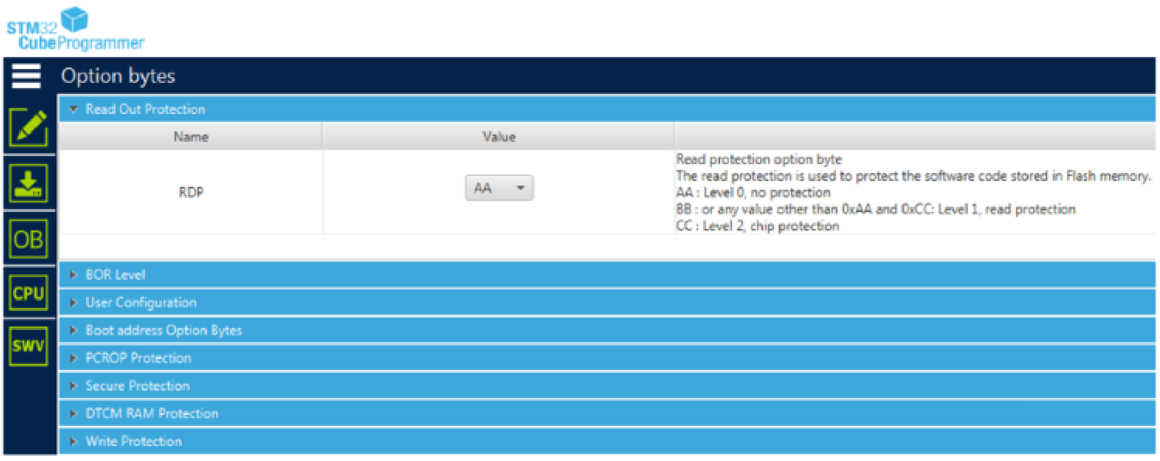
Connect to the board using STM32CubeProgrammer and follow these steps:

1. Check the RDP option byte: must be set to AA.
2. Check the security bit: must be set to 0.
3. Remove the complete Flash memory content.

2.1.1 RDP option byte

In the STM32CubeProgrammer, click on *OB*, and *Read Out Protection*: check the RDP option byte value, that must be AA (Level 0). If the RDP is set to BB (Level 1), select AA and click on *Apply*.

Figure 4. RDP option byte



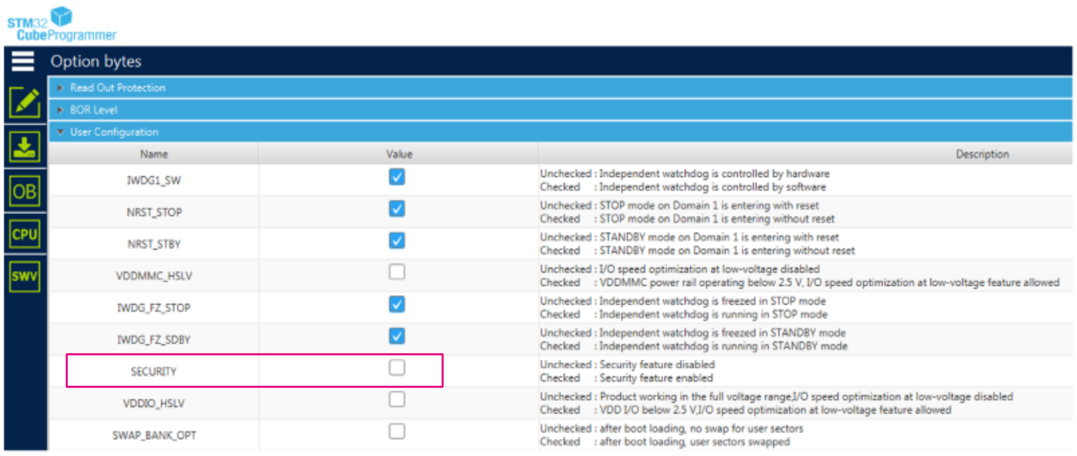
Name	Value	
RDP	AA	Read protection option byte The read protection is used to protect the software code stored in Flash memory. AA : Level 0, no protection BB : or any value other than 0xAA and 0xCC: Level 1, read protection CC : Level 2, chip protection

It resets the RDP to Level 0 and the Flash memory content is erased.

The debugger is lost and a re-connection to the STM32CubeProgrammer is needed.

2.1.2 Security bit

In the STM32CubeProgrammer, click on *OB* and *User Configuration*: check the security option byte. *SECURITY* must be unchecked (security feature disabled). If the security option bit is set, uncheck *SECURITY* and click on *Apply*.

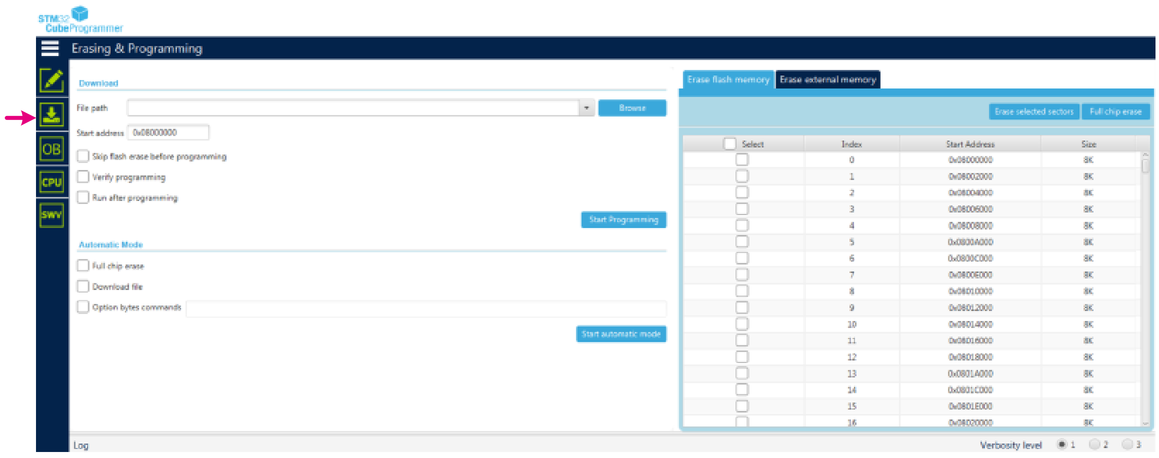


Name	Value	Description
IWDG1_SW	✓	Unchecked : Independent watchdog is controlled by hardware Checked : Independent watchdog is controlled by software
NRST_STOP	✓	Unchecked : STOP mode on Domain 1 is entering with reset Checked : STOP mode on Domain 1 is entering without reset
NRST_STBY	✓	Unchecked : STANDBY mode on Domain 1 is entering with reset Checked : STANDBY mode on Domain 1 is entering without reset
VDDMMC_HSLV	☐	Unchecked : I/O speed optimization at low-voltage disabled Checked : VDDMMC power rail operating below 2.5 V, I/O speed optimization at low-voltage feature allowed
IWDG_FZ_STOP	✓	Unchecked : Independent watchdog is freezed in STOP mode Checked : Independent watchdog is running in STOP mode
IWDG_FZ_SDBY	✓	Unchecked : Independent watchdog is freezed in STANDBY mode Checked : Independent watchdog is running in STANDBY mode
SECURITY	☐	Unchecked : Security feature disabled Checked : Security feature enabled
VDDIO_HSLV	☐	Unchecked : Product working in the full voltage range,I/O speed optimization at low-voltage disabled Checked : VDD I/O below 2.5 V,I/O speed optimization at low-voltage feature allowed
SWAP_BANK_OPT	☐	Unchecked : after boot loading, no swap for user sectors Checked : after boot loading, user sectors swapped

Figure 5. Security option bit

2.1.3 Erase the current Flash memory content

If the Flash memory is not empty, it is advised to erase the Flash memory before starting the tutorial. In the STM32CubeProgrammer, click on *Erasing & Programming* and on *Full chip erase*.



Start address: 0x08000000

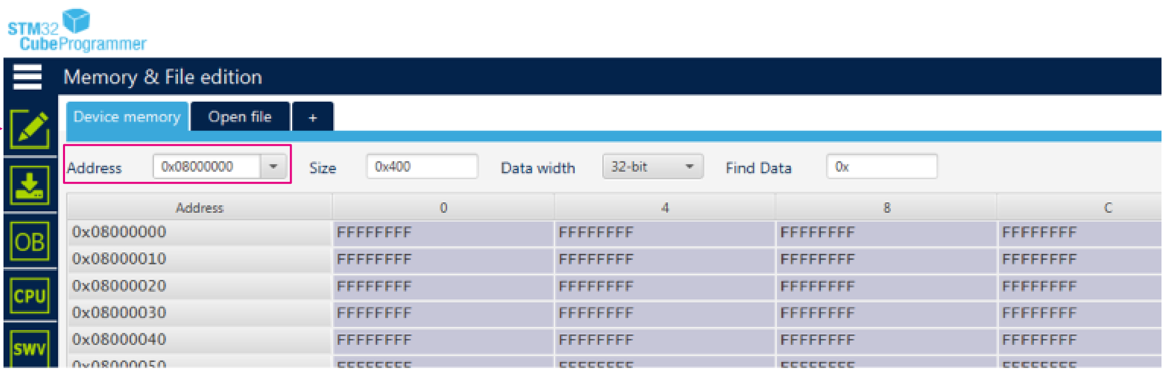
Select	Index	Start Address	Size
☐	0	0x08000000	8K
☐	1	0x08002000	8K
☐	2	0x08004000	8K
☐	3	0x08006000	8K
☐	4	0x08008000	8K
☐	5	0x0800A000	8K
☐	6	0x0800C000	8K
☐	7	0x0800E000	8K
☐	8	0x08010000	8K
☐	9	0x08012000	8K
☐	10	0x08014000	8K
☐	11	0x08016000	8K
☐	12	0x08018000	8K
☐	13	0x0801A000	8K
☐	14	0x0801C000	8K
☐	15	0x0801E000	8K
☐	16	0x08020000	8K

Verbosity level: [✓ 1] 2 3

Figure 6. Full chip erase

Then click on *Memory & file edition*. The Flash memory content is erased and displayed as FFFFFFFF.

Figure 7. **Read of Flash memory content**



Address 0x08000000 Size 0x400 Data width 32-bit Find Data 0x

Address	0	4	8	C
0x08000000	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x08000010	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x08000020	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x08000030	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x08000040	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF

Read also the content of the addresses starting at 0x08100000, corresponding to the second Flash bank that must also be completely erase.

2.2 Step 2: compile and load the two firmwares

The firmwares are available in the `Applications` directory of the STM32Cube_FW_H7.

The HDP directory contains the two firmwares, HDP_Boot and HDP_Appli, that needs to be flashed separately.

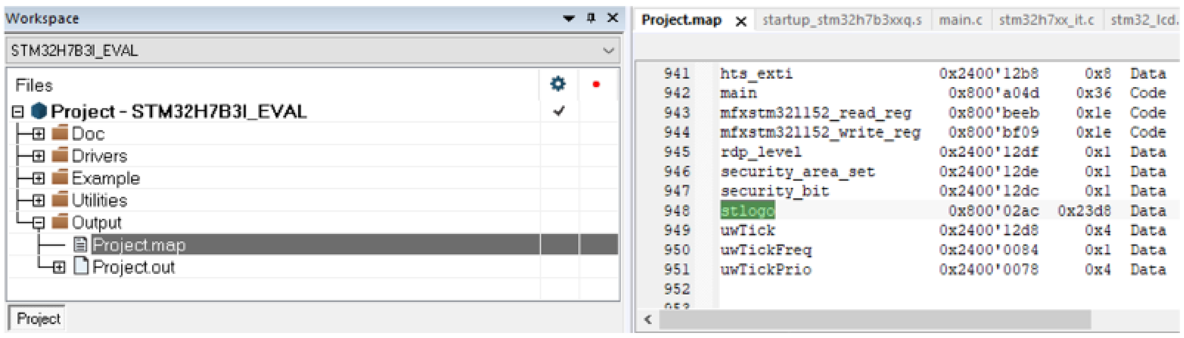
2.2.1 HDP_Boot

Open the HDP_Boot firmware with one of the supported tool chain. Compile and load the code. This first firmware can be tried out, running step by step. The code is executed starting at Flash memory address 0x08000000. Explanations appear on the display.

The HDP_Boot is located in a Flash memory address range that is set later as a secure area.

Note: *The stlogo is stored in the first Flash bank. See IAR Embedded Workbench® example below, showing that the stlogo is stored in a Flash memory area that is set later as secure*

*Figure 8. **First stlogo stored in Flash bank 1***



```
hts_exti                  0x2400'12b8     0x8  Data
main                       0x800'a04d    0x36  Code
mfxstm321152_read_reg      0x800'beeb    0x1e  Code
mfxstm321152_write_reg     0x800'bf09    0x1e  Code
rdp_level                 0x2400'12df     0x1  Data
security_area_set         0x2400'12de     0x1  Data
security_bit              0x2400'12dc     0x1  Data
stlogo                     0x800'02ac  0x23d8  Data
uwTick                    0x2400'12d8     0x4  Data
uwTickFreq                0x2400'0084     0x1  Data
uwTickPrio                0x2400'0078     0x4  Data
```

Check in the `main.c`, the instruction `exitSecureArea`, that calls a service of the RSS described in the product reference manual:

```
RSS->exitSecureArea((uint32_t) 0x08100000, RSS_ENABLE_JTAG_AT_EXIT);
```

This code jumps to the defined address (0x08100000) where the firmware HDP_Appli is loaded.

In this example, the start address of the second Flash bank has been chosen, but the user can define any address including an address located in the first Flash bank.

The user must ensure that HDP_Boot and HDP_Appli firmwares do not have overlapping address ranges, and that both compiled code fit in the selected ranges.

2.2.2 HDP_Appli

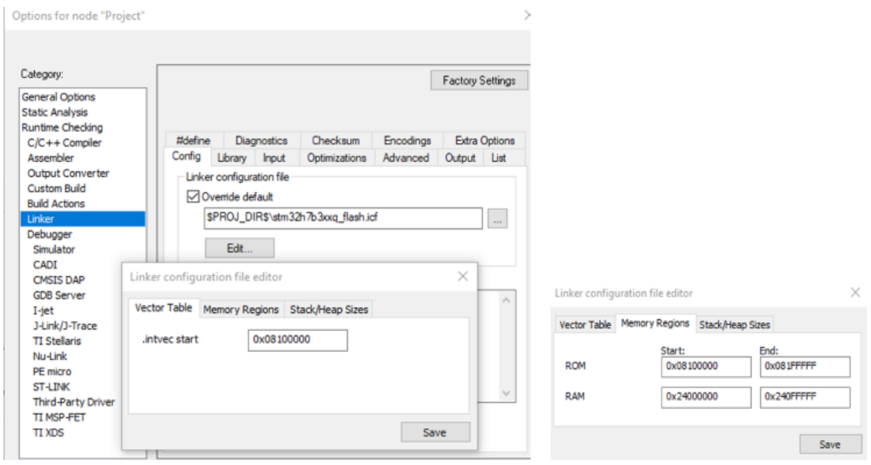
Open the HDP_Appli firmware with one of the supported tool chain and check the linker option.

For IAR, go to *Linker -> Config -> Edit*, then *-> Vector Table* or *-> Memory Regions*.

Note: *The vector table and the ROM have been relocated to the address 0x08100000 that corresponds to the second Flash bank.*

The IAR setting is shown in the figure below.

Figure 9. IAR setting for code loaded in Flash bank 2



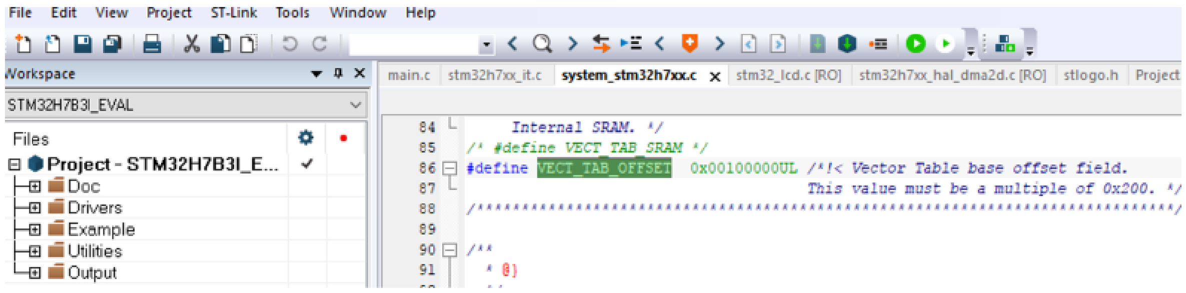
Linker configuration file
☑ Override default
$PROJ_DIR$\stm32h7b3xxq_flash.icf

.intvec start	0x08100000	

	Start:	End:
ROM	0x08100000	0x081FFFFF
RAM	0x24000000	0x240FFFFF

Note: *The setting in the linker option is not sufficient to relocate the vector table. The correct offset has to be defined in the* `system_stm32h7xx.c` *file. See the IAR example in the figure below.*

Figure 10. Vector table relocation to Flash bank 2 (IAR example)



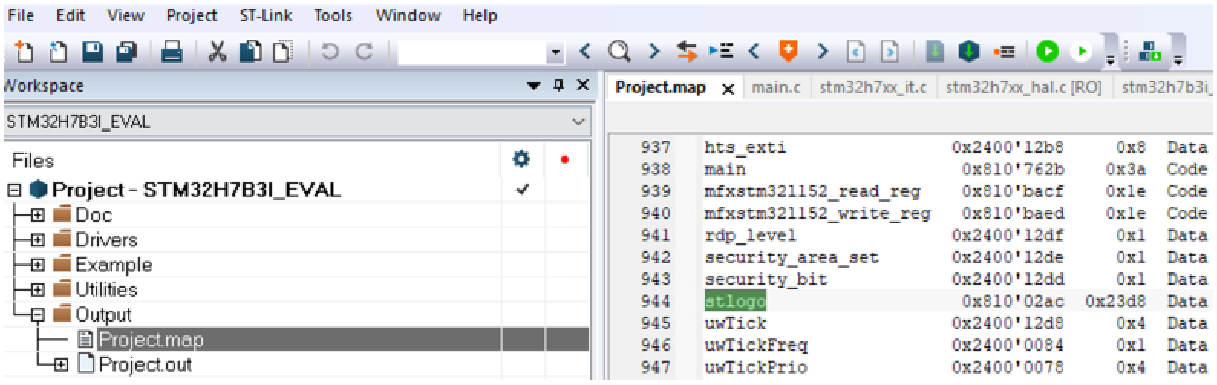
```
     Internal SRAM. */
/* #define VECT_TAB_SRAM */
#define VECT_TAB_OFFSET  0x00100000UL /*!< Vector Table base offset field.
                                     This value must be a multiple of 0x200. */
/*******************************************************************************/

/**
  * @}
```

Compile and load the HDP_Appli. The second firmware can be tried out, running step by step. The code is executed starting at Flash address 0x08100000. Explanations appear on the display

Note: *The stlogo is stored in the second Flash bank. The same image is now stored twice: one in the first and one in the second Flash bank.*

The IAR example in the figure below shows that the stlogo is stored in a Flash memory area that stays non-secure.

Figure 11. **Second stlogo stored in Flash bank 2**



```
hts_exti                  0x2400'12b8      0x8  Data
main                       0x810'762b     0x3a  Code
mfxstm321152_read_reg      0x810'bacf     0x1e  Code
mfxstm321152_write_reg     0x810'baed     0x1e  Code
rdp_level                 0x2400'12df      0x1  Data
security_area_set         0x2400'12de      0x1  Data
security_bit              0x2400'12dd      0x1  Data
stlogo                     0x810'02ac   0x23d8  Data
uwTick                    0x2400'12d8      0x4  Data
uwTickFreq                0x2400'0084      0x1  Data
uwTickPrio                0x2400'0078      0x4  Data
```

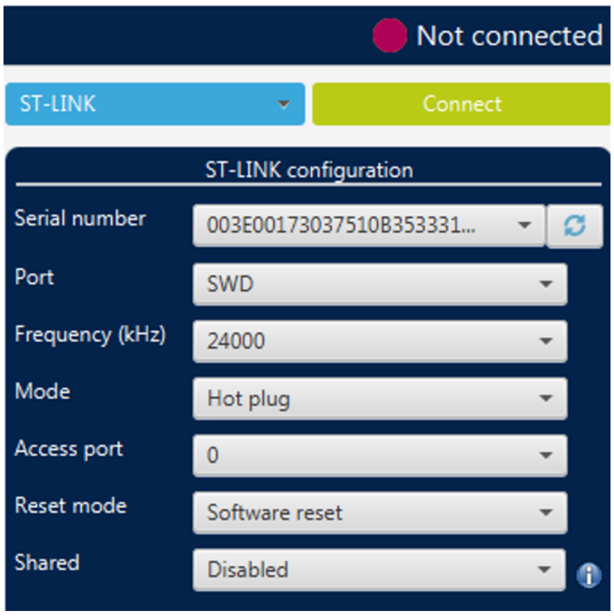
The debugger can now be disconnected and the tool chain closed.

2.3 Step 3: standalone execution

Proceed with a power-down reset, the application starts standalone. Ensure both firmwares are executed correctly according to the indications on the display.

At the end of the second firmware execution, a connection to STM32CubeProgrammer is required. See the settings (shown in the figure below) to select in STM32CubeProgrammer before clicking *Connect* (the product serial number is automatically filled in). Since a code is currently executed, the connecting mode is *Hot plug*.

Figure 12. **Connection to STM32CubeProgrammer**

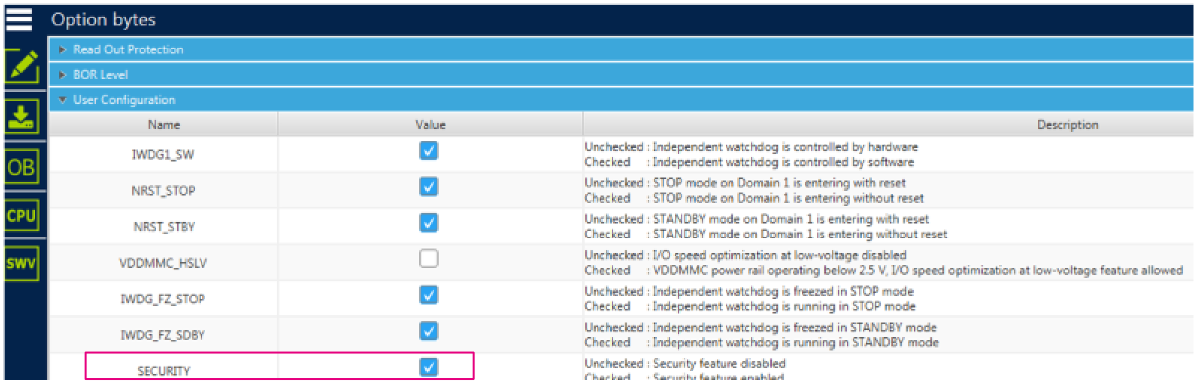


Serial number	003E00173037510B353331...
Port	SWD
Frequency (kHz)	24000
Mode	Hot plug
Access port	0
Reset mode	Software reset
Shared	Disabled

2.3.1 Security option bit setting

Set the security bit using STM32CubeProgrammer, as shown below: Click on *OB* and *User Configuration*.
Click on *SECURITY* and *Apply*.

Figure 13. **Security bit setting with STM32CubeProgrammer**



Name	Value	Description
IWDG1_SW	✓	Unchecked : Independent watchdog is controlled by hardware Checked : Independent watchdog is controlled by software
NRST_STOP	✓	Unchecked : STOP mode on Domain 1 is entering with reset Checked : STOP mode on Domain 1 is entering without reset
NRST_STBY	✓	Unchecked : STANDBY mode on Domain 1 is entering with reset Checked : STANDBY mode on Domain 1 is entering without reset
VDDMMC_HSLV	☐	Unchecked : I/O speed optimization at low-voltage disabled Checked : VDDMMC power rail operating below 2.5 V, I/O speed optimization at low-voltage feature allowed
IWDG_FZ_STOP	✓	Unchecked : Independent watchdog is freezed in STOP mode Checked : Independent watchdog is running in STOP mode
IWDG_FZ_SDBY	✓	Unchecked : Independent watchdog is freezed in STANDBY mode Checked : Independent watchdog is running in STANDBY mode
SECURITY	✓	Unchecked : Security feature disabled Checked : Security feature enabled

Click on *Memory & File edition* and read the Flash memory content, that starts at address 0x08000000 and at address 0x08100000. Both stored codes are visible.

Proceed with a power-down reset and verify that the two firmwares are still executed correctly.

Important: *Do not bypass any steps described in this tutorial.*

2.3.2 Secure area setting

Set the secure area using STM32CubeProgrammer, as shown below.

Important: *Before setting a secure address range, it is very important to have a valid code in the secure area.*

After booting, the code stored in the secure area is automatically executed. During this code execution, the debugger cannot be connected, it is like an extension of the boot but with user defined code.

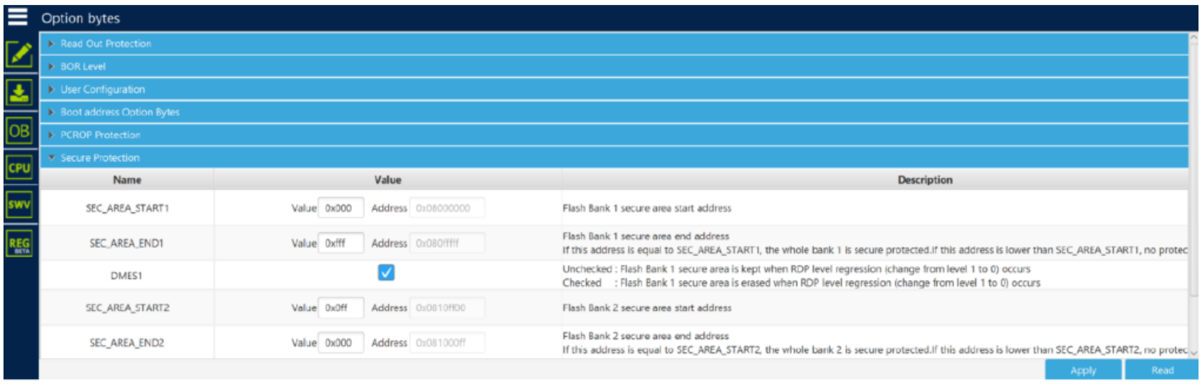
The last instruction executed in this secure area is a call to a service of the RSS for a “jump” out of the secure area (RSS->exitSecureArea).

The debugger can then be connected, but the Flash memory content of the chosen secure area is read out as 00000000.

If, for any reason, the execution in the secure area is locked and the “jump” instruction is never executed, it is not possible anymore to connect a debugger and also not possible anymore to modify the Flash memory content.

The message on the evaluation board display indicates the start and end address to set.

In the STM32CubeProgrammer, click on *OB* and *Secure Protection*.



Option bytes

- Read Out Protection
- BOR Level
- User Configuration
- Boot address Option Bytes
- PCROP Protection
- Secure Protection

Name	Value	Description
SEC_AREA_START1	Value 0x000 Address 0x08000000	Flash Bank 1 secure area start address
SEC_AREA_END1	Value 0xfff Address 0x080ffff	Flash Bank 1 secure area end address If this address is equal to SEC_AREA_START1, the whole bank 1 is secure protected.If this address is lower than SEC_AREA_START1, no protec
DMES1	✓	Unchecked : Flash Bank 1 secure area is kept when RDP level regression (change from level 1 to 0) occurs Checked : Flash Bank 1 secure area is erased when RDP level regression (change from level 1 to 0) occurs
SEC_AREA_START2	Value 0x0ff Address 0x0810ff00	Flash Bank 2 secure area start address
SEC_AREA_END2	Value 0x000 Address 0x081000ff	Flash Bank 2 secure area end address If this address is equal to SEC_AREA_START2, the whole bank 2 is secure protected.If this address is lower than SEC_AREA_START2, no protec

Figure 14. **Secure area setting with STM32CubeProgrammer**

For this example, the wanted start address is 0x08000000 and the end address is 0x080FFFF0.

The addresses are entered through the value fields containing three digits. These fields correspond to FLASH_SCAR_CUR1 and FLASH_SCAR_CUR2 registers (see the product reference manual).

The related start and end addresses are automatically filled:

- Entering 0x000 as starting value gives the wanted secure area starting at the beginning of Flash bank 1.
- Entering 0xFFF as end value gives a secure area end address of 0x080FFFFF.
- Entering 0xFFE as end value gives a secure area end address of 0x080FFEFF.

Note: *The wanted end value of 0x080FFFF0 is not possible.*

For this example, all above values work, but read carefully the following explanations.

A secure area can only be set by blocks of 256 bytes (0x100). The start address is the address of the beginning of the first secure area block of 256 bytes. The end address is the address of the beginning of the last secure area block of 256 bytes:

- With Start1 value yyy, the secure area begins at 0x080yyy00.
- With End1 value zzz, the secure area ends at 0x080zzzFF.

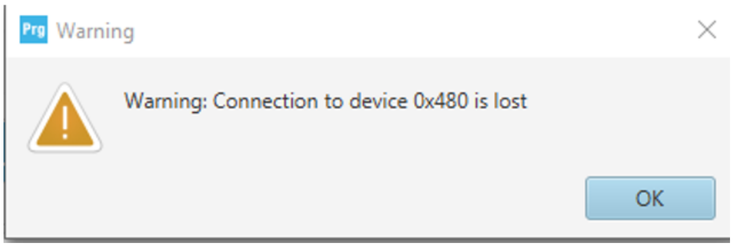
In other words:

- The first secured block of 256 bytes starts at 0x080yyy00 and ends at 0x080yyyFF.
- The last secured block of 256 bytes starts at 0x080zzz00 and ends at 0x080zzzFF.

The consequence of the previous explanations is important to note, specially for highly optimized code sizes: **the end address of the secure code cannot be in the same 256-byte block than the start address of the non-secure code.**

Note: *If the entered start and end values are identical, the complete related Flash bank is set as a secure area.*

After clicking on *Apply*, the secure area is set and the device reboots automatically. The debugger is disconnected as expected and the following message is displayed.



Warning

Warning: Connection to device 0x480 is lost

Figure 15. **STM32CubeProgrammer message after setting a secure area**

2.3.3 Standalone execution with secure-area set

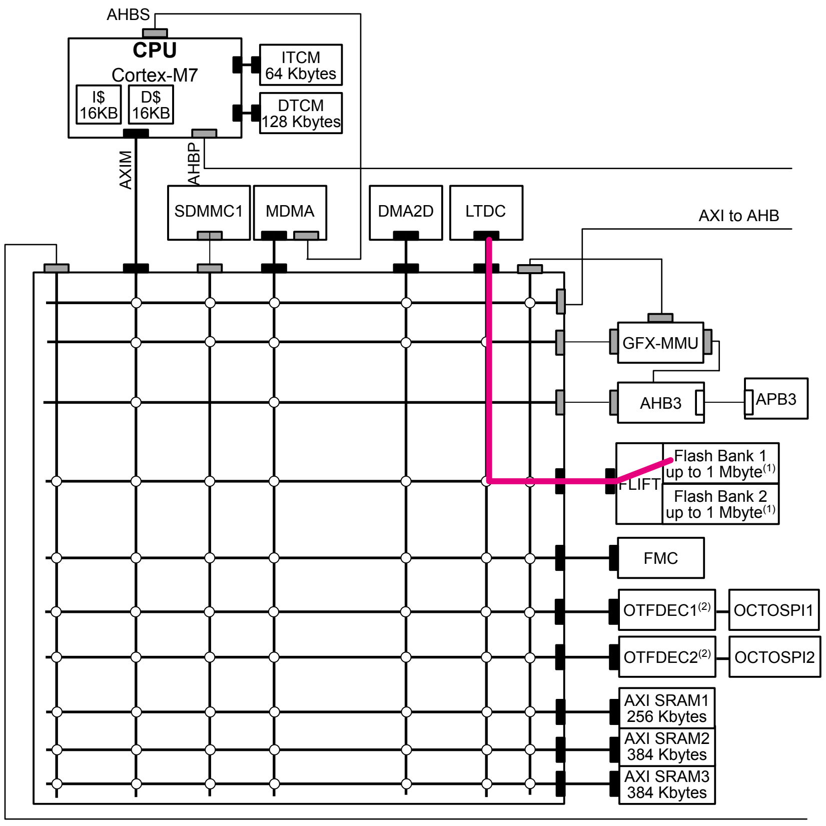
Proceed with a power-down reset, the first firmware is now executed in a secure area. Follow the indications of the display.

As indicated on the display, the stlogo is displayed as a black box. The stlogo of the first firmware is now stored in a secure area. Any attempt to read data stored in the secure area is not executed and 00000000 is returned.

This is the reason the stlogo is now displayed as a black box and was displayed correctly before the secure area had been set.

The following schematic shows the path between the LTDC and the Flash memory.

Figure 16. STM32H7B3xx AXI matrix



Pursue the code execution. When the second firmware in the non-secure area is executed, the debugger can be reconnected.

Connect to the STM32CubeProgrammer. The Flash memory in the secured area is displayed as 00000000. The content of the non-secure Flash memory is seen as previously. The stlogo is again displayed since the stlogo stored in the Flash bank 2 (non secured) has been loaded.

Continue the code execution till the end.

2.4 Step 4: removal of the secure area

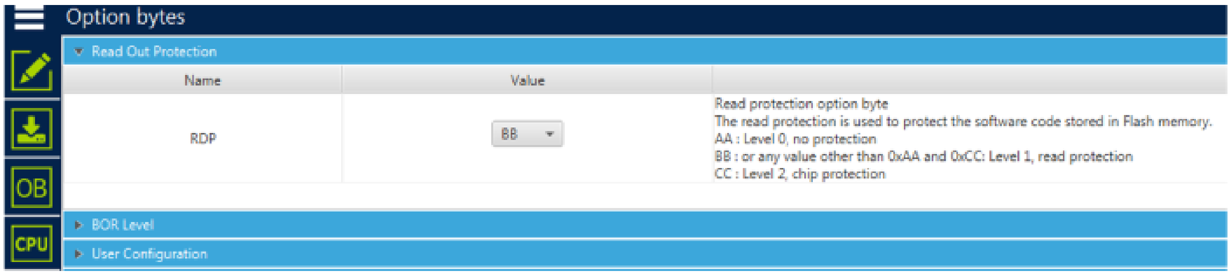
To remove the secure area, first the readout protection (RDP) must be set to Level 1. A regression to Level 0 insures the complete removal of the Flash memory content (see the product reference manual for the RDP description).

Note: *Take care to not set the readout protection to Level 2 that definitively locks the device.*

2.4.1 Setting RDP to level 1

In the STM32CubeProgrammer, click on *OB* and *Read Out Protection*, select *BB* to set to Level 1 and click on *Apply*.

Figure 17. **RDP Level 1 setting in STM32CubeProgrammer**



RDP: BB

Read protection option byte
The read protection is used to protect the software code stored in Flash memory.
AA : Level 0, no protection
BB : or any value other than 0xAA and 0xCC: Level 1, read protection
CC : Level 2, chip protection

As expected, the debugger connection is lost and the same message than in Figure 15 is displayed. Proceed with a power-down reset, the application restarts standalone. The application is now executed with the readout protection level set to 1.

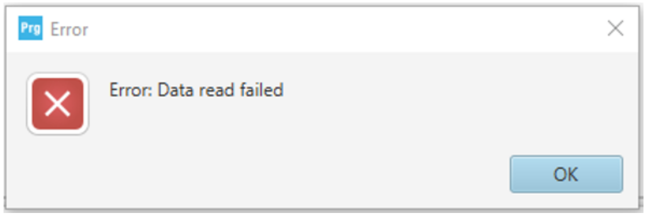
Execute the firmware till the end.

2.4.2 Resetting the secure area and Flash memory content removal

Proceed with a power-down reset, the application restarts standalone. Before connecting the STM32CubeProgrammer, the second firmware must be executed in order to have "jumped" out of the secure area execution.

In the STM32CubeProgrammer, click on *Connect* (*Hot plug* selected as mode). The message in the figure below is displayed since the Flash memory content cannot be read due to the RDP Level 1.

Figure 18. **Flash memory content not readable in RDP Level 1**



Error: Data read failed

But the option byte can be accessed.

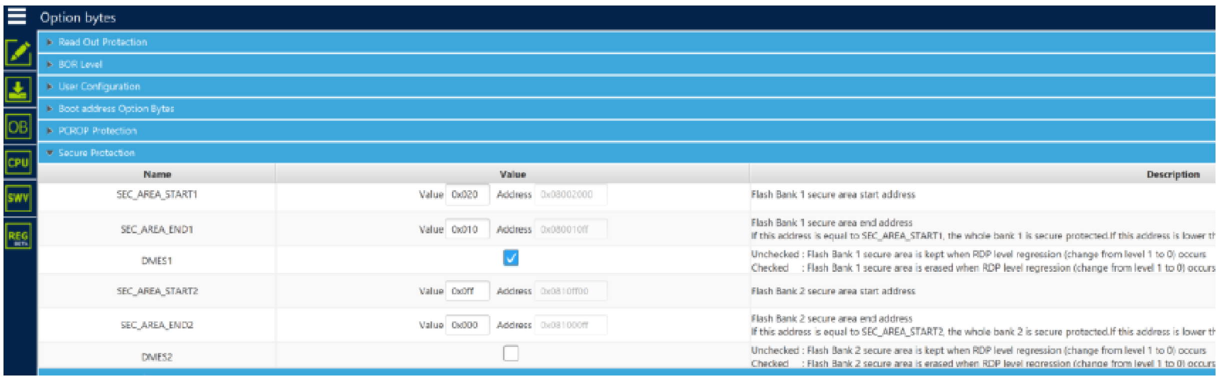
In the STM32CubeProgrammer: Click on *OB* and open *Read Out Protection* and *Secure Protection* windows.

Set RDP to AA. **!! Important: DO NOT click on *Apply*** before having set the secure area range and having chosen the DMES bit.

Set *secure area start address* > *secure area end address* (It means no secure area is set).

The figure below shows a proposed area address setting to perform a regression.

Click *Apply* only when all settings are defined as expected.



Name	Value	Description
SEC_AREA_START1	Value 0x020 Address 0x08002000	Flash Bank 1 secure area start address
SEC_AREA_END1	Value 0x010 Address 0x080010ff	Flash Bank 1 secure area end address If this address is equal to SEC_AREA_START1, the whole bank 1 is secure protected.If this address is lower th
DMES1	✓	Unchecked : Flash Bank 1 secure area is kept when RDP level regression (change from level 1 to 0) occurs Checked : Flash Bank 1 secure area is erased when RDP level regression (change from level 1 to 0) occurs
SEC_AREA_START2	Value 0x0ff Address 0x0810ff00	Flash Bank 2 secure area start address
SEC_AREA_END2	Value 0x000 Address 0x081000ff	Flash Bank 2 secure area end address If this address is equal to SEC_AREA_START2, the whole bank 2 is secure protected.If this address is lower th
DMES2	☐	Unchecked : Flash Bank 2 secure area is kept when RDP level regression (change from level 1 to 0) occurs Checked : Flash Bank 2 secure area is erased when RDP level regression (change from level 1 to 0) occurs

Figure 19. **Removal of a secure area with STM32CubeProgrammer**

Important:

For this tutorial, the DMES option bit must be checked.

In this example, the secure code and the non-secure code or both stored in Flash memory. If the DMES is unchecked the secure area and content is kept but the non-secure code erased through the regression. After the device reboots, the secure code is executed but the jump-out of the secure area is to an empty Flash memory address creating a HardFault.

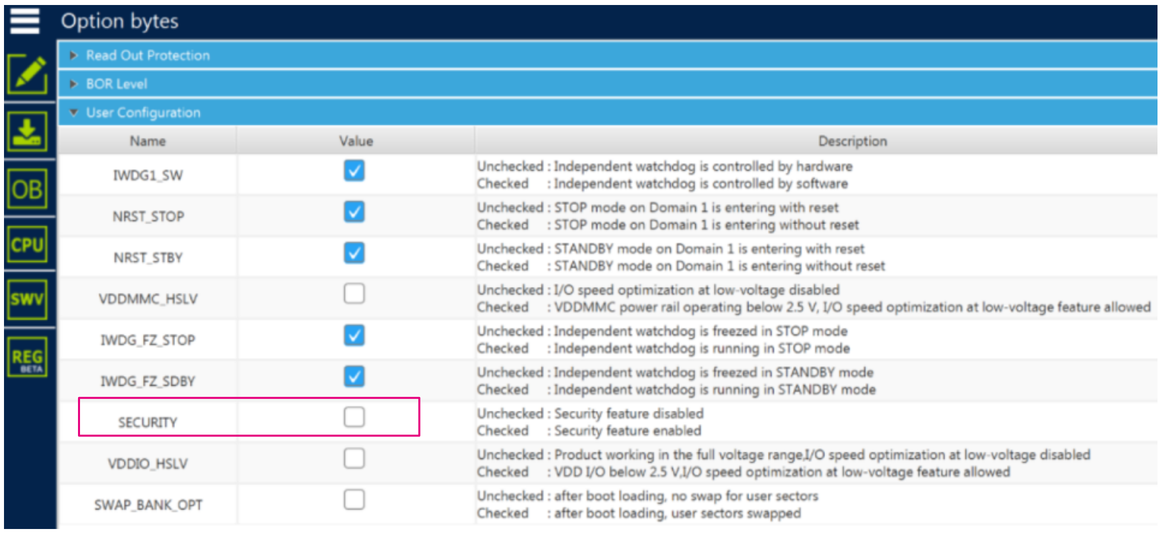
It is not anymore possible to connect a debugger, since with a remaining secure area, at power up, the unique boot entry is through the RSS, and the secure code is automatically executed.

If the DMES bit is unchecked, the secure code must insure a jump-out of the secure area to an address containing an application code still valid after the regression.

The full Flash memory content is now erased and the secure area is removed.

2.4.3 Resetting the secure option bit

The last step is to remove the security bit. In STM32CubeProgrammer: Click on *OB* and *User Configuration*. Uncheck *SECURITY* and click on *Apply*.



Name	Value	Description
IWDG1_SW	✓	Unchecked : Independent watchdog is controlled by hardware Checked : Independent watchdog is controlled by software
NRST_STOP	✓	Unchecked : STOP mode on Domain 1 is entering with reset Checked : STOP mode on Domain 1 is entering without reset
NRST_STBY	✓	Unchecked : STANDBY mode on Domain 1 is entering with reset Checked : STANDBY mode on Domain 1 is entering without reset
VDDMMC_HSLV	☐	Unchecked : I/O speed optimization at low-voltage disabled Checked : VDDMMC power rail operating below 2.5 V, I/O speed optimization at low-voltage feature allowed
IWDG_FZ_STOP	✓	Unchecked : Independent watchdog is freezed in STOP mode Checked : Independent watchdog is running in STOP mode
IWDG_FZ_SDBY	✓	Unchecked : Independent watchdog is freezed in STANDBY mode Checked : Independent watchdog is running in STANDBY mode
SECURITY	☐	Unchecked : Security feature disabled Checked : Security feature enabled
VDDIO_HSLV	☐	Unchecked : Product working in the full voltage range,I/O speed optimization at low-voltage disabled Checked : VDD I/O below 2.5 V,I/O speed optimization at low-voltage feature allowed
SWAP_BANK_OPT	☐	Unchecked : after boot loading, no swap for user sectors Checked : after boot loading, user sectors swapped

Figure 20. **Security option bit reset with STM32CubeProgrammer**

3 Conclusion

This application note explains the secure-area concept also called secure hide protection (HDP).

The provided practical example clarifies how to set, use and remove a secure area for the embedded Flash memory.

It is important to follow carefully every step, even for user developments, since, in some explained cases, the device can be locked and cannot be retrieved.

Before setting the secure area, it is advised to extensively test the firmware intended to be executed in this area. All the possible execution locks must be treated.

Revision history

Table 2. **Document revision history**

Date	Version	Changes
10-Feb-2021	1	Initial release.

Contents

List of figures