

ACTIVATE SECURE MEMORY

Link

<https://www.youtube.com/watch?v=qCuVBD2dmTA&list=PLnMKNibPkDnFzux3PHKUEi14ftDn9Cbm7&index=12>

Description

In this paperwork, we will configure the secure memory, and then activate it to see how to protect some portions of code.

Contents

Link	1
Description	1
Prerequisites	2
STM32 Board.....	2
ST-Link cable	2
STM32CubeProgrammer	2
STM32CubeMX	2
STM32CubeIDE.....	2
Walkthrough	3
Step 1 : Launch STM32CubeMX and generate the code	3
Step 2 : Create The test code	4
Step 3 : Separate the memory	5
Step 4 : Activate the secure memory	6
Step 5 : Check the secure memory	7

Prerequisites

Security Features by STM32 Series 7

STM32 Series	Security Features																Arm Cortex®
	96-Bit Unique ID	FLASH WRP	FLASH PCROP	FLASH RDP	Unique entry point	Secure mem/HDP	MPU	Firewall	Trustzone	OTFDEC	Tamper	TRNG	CRYPT AES	HASH	PKA	CryptoLib	
STM32 F0	■	■		■							■					■	M0
STM32 F1	■	■					■				■					■	M3
STM32 F2	■	■		■			■				■	■	■	■		■	M3
STM32 F3	■	■		■			■				■					■	M4
STM32 F4	■	■	■	■			■				■	■	■	■		■	M4
STM32 F7	■	■	■	■			■				■	■	■	■		■	M7
STM32 L0	■	■		■			■	■			■		■			■	M0+
STM32 L1	■	■		■			■				■		■			■	M3
STM32 L4	■	■		■			■	■			■	■	■			■	M4
STM32 L5	■	■		■	■	■	■		■	■	■	■	■	■	■	■	M33
STM32 H7	■	■	■	■	■	■	■				■	■	■	■		■	M7/M4
STM32 G0	■	■		■		■	■				■		■			■	M0+
STM32 G4	■	■		■		■	■				■		■			■	M4
STM32 WB	■	■		■			■				■		■		■	■	M4/M0+

Available on all devices

Depends on device part number

STM32 Board

ST-Link cable

STM32CubeProgrammer

STM32CubeMX

STM32CubeIDE

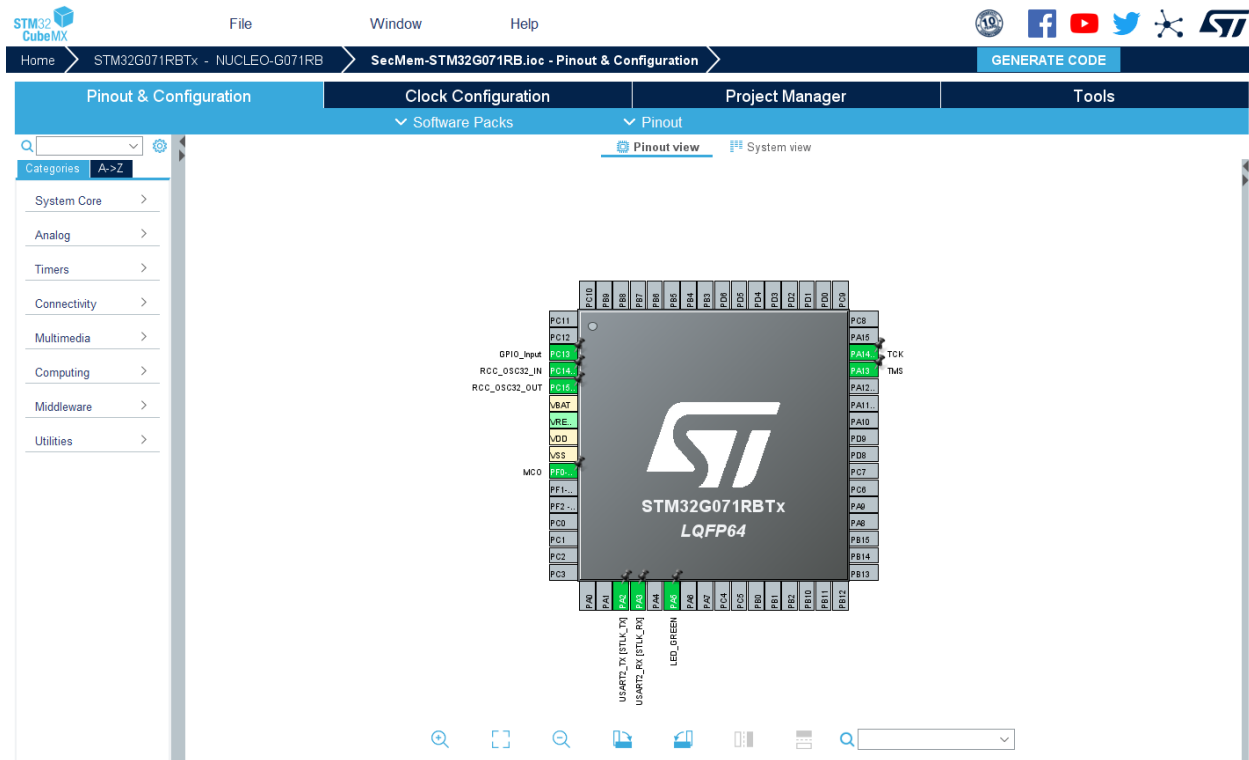
Walkthrough

Step 1 : Launch STM32CubeMX and generate the code

Launch STM32CubeMX and select the right board depending on the one you are using. In my case I use the G071RB Nucleo board. Then you can generate the code of your project.

Don't forget to select the correct IDE (in my case STM32CubeIDE).

Please make sure that the PC13 is set as a GPIO_input because we want to use it as a push button.

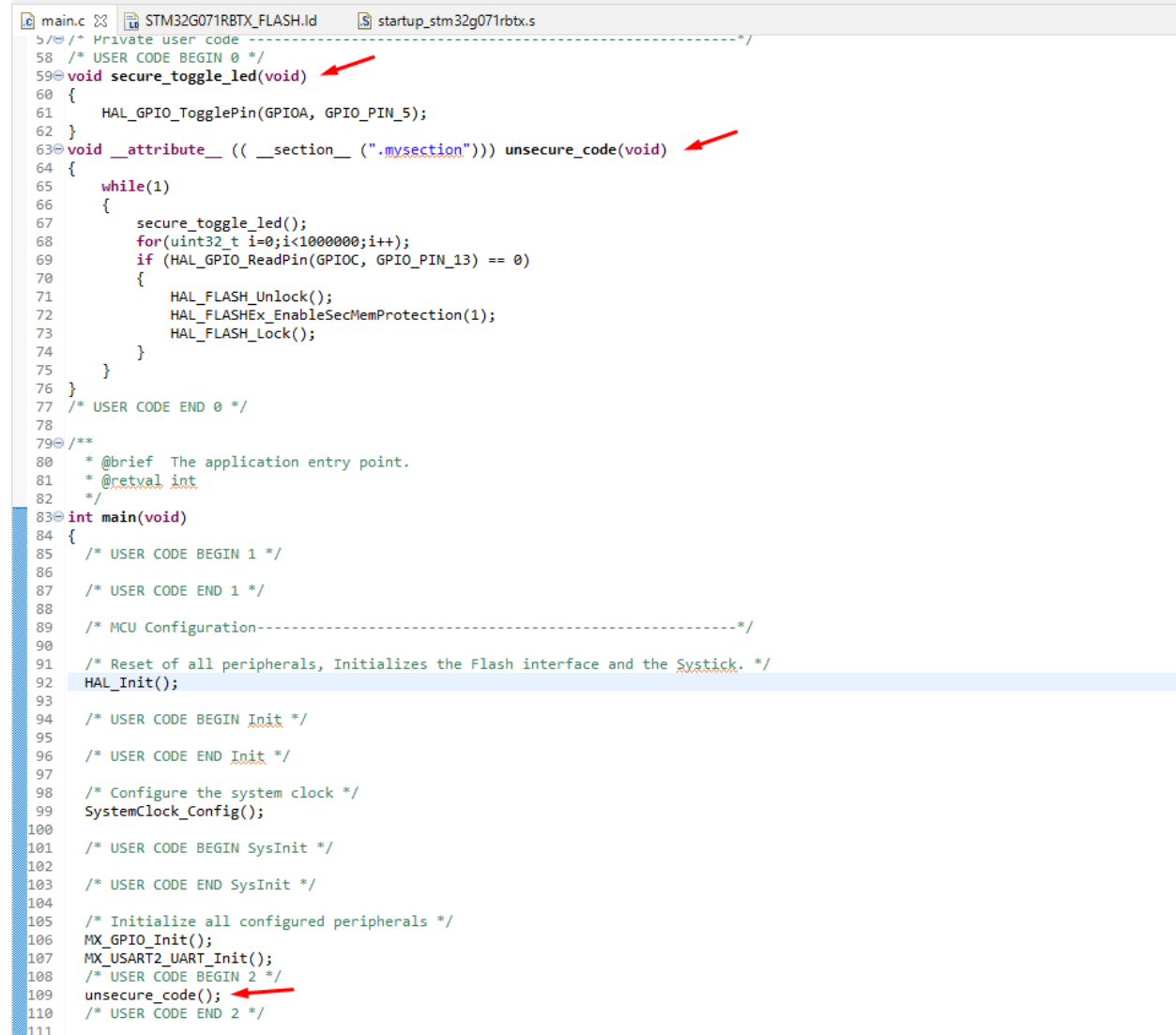


Step 2 : Create The test code

Basically we call the `unsecure_code` function in the main.

This function permits to call the `secure_toggle_led` function and to enter in the secure memory protection if the button is pressed.

The section attribute will link our part of memory to our function.



```

main.c STM32G071RBTX_FLASH.ld startup_stm32g071rbtx.s
57 /* Private user code ----- */
58 /* USER CODE BEGIN 0 */
59 void secure_toggle_led(void)
60 {
61     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
62 }
63 void __attribute__((section(".mysection"))) unsecure_code(void)
64 {
65     while(1)
66     {
67         secure_toggle_led();
68         for(uint32_t i=0; i<1000000; i++);
69         if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == 0)
70         {
71             HAL_FLASH_Unlock();
72             HAL_FLASHEx_EnableSecMemProtection(1);
73             HAL_FLASH_Lock();
74         }
75     }
76 }
77 /* USER CODE END 0 */
78
79 /**
80  * @brief The application entry point.
81  * @retval int
82  */
83 int main(void)
84 {
85     /* USER CODE BEGIN 1 */
86
87     /* USER CODE END 1 */
88
89     /* MCU Configuration----- */
90
91     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
92     HAL_Init();
93
94     /* USER CODE BEGIN Init */
95
96     /* USER CODE END Init */
97
98     /* Configure the system clock */
99     SystemClock_Config();
100
101     /* USER CODE BEGIN SysInit */
102
103     /* USER CODE END SysInit */
104
105     /* Initialize all configured peripherals */
106     MX_GPIO_Init();
107     MX_USART2_UART_Init();
108     /* USER CODE BEGIN 2 */
109     unsecure_code();
110     /* USER CODE END 2 */
111

```

Step 3 : Separate the memory

Now in the FLASH.id you have to create the new section of memory. This one will permit us to put the unsecure code only in FLASH_unsecure memory.

To do so we add a section .mysection that corresponds to the FLASH_unsecure we created.

```

main.c  STM32G071RBTX_FLASH.ld  startup_stm32g071rbtx.s  STM32G071RBTX_FLASH.ld
34 /*
35
36 /* Entry Point */
37 ENTRY(Reset_Handler)
38
39 /* Highest address of the user mode stack */
40 _estack = ORIGIN(RAM) + LENGTH(RAM); /* end of "RAM" Ram type memory */
41
42 _Min_Heap_Size = 0x200; /* required amount of heap */
43 _Min_Stack_Size = 0x400; /* required amount of stack */
44
45 /* Memories definition */
46 MEMORY
47 {
48   RAM      (xrw)  : ORIGIN = 0x20000000, LENGTH = 36K
49   FLASH    (rx)   : ORIGIN = 0x80000000, LENGTH = 120K
50   FLASH_unsecure(rx): ORIGIN = 0x801E0000, LENGTH = 8K
51 }
52
53 /* Sections */
54 SECTIONS
55 {
56   .mysection :
57   {
58     . = ALIGN(4);
59     *(.mysection)
60     . = ALIGN(4);
61   } > FLASH_unsecure

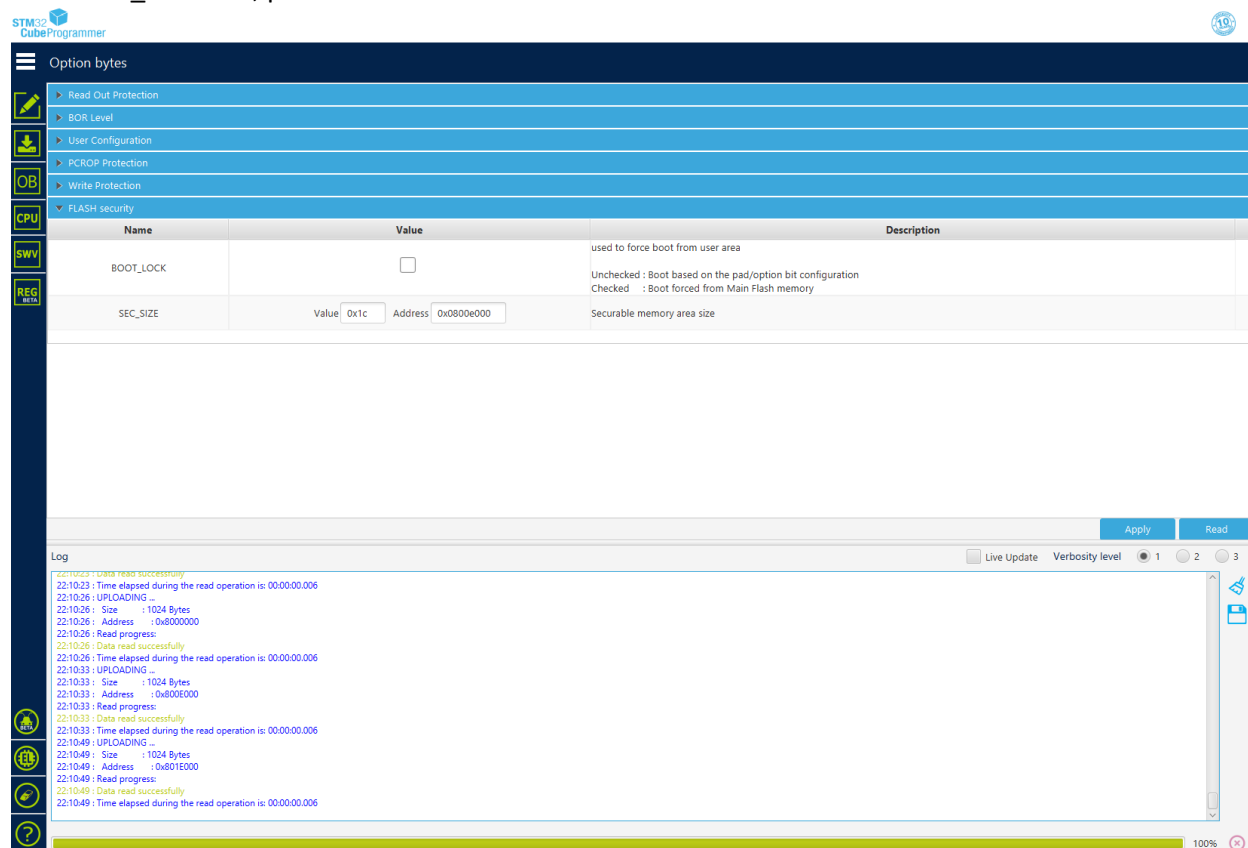
```

When you have finished you can compile the code and execute the debugger.

Step 4 : Activate the secure memory

Launch the STM32Programmer and go to OB and then FLASH security.

In the SEC_SIZE field, put as a value 0x1c.



The screenshot shows the STM32CubeProgrammer interface. The 'Option bytes' section is expanded, and the 'FLASH security' tab is selected. The 'SEC_SIZE' field is set to '0x1c'. The 'Value' field is '0x1c' and the 'Address' field is '0x0800e000'. The 'Description' for SEC_SIZE is 'Securable memory area size'. The 'BOOT_LOCK' field is set to '0' (unchecked). The 'Description' for BOOT_LOCK is 'used to force boot from user area'. The 'Log' window at the bottom shows the progress of the operation, indicating that the data was read successfully and the operation is complete.

Name	Value	Description
BOOT_LOCK	<input type="checkbox"/>	used to force boot from user area Unchecked : Boot based on the pad/option bit configuration Checked : Boot forced from Main Flash memory
SEC_SIZE	Value: 0x1c Address: 0x0800e000	Securable memory area size

Log

22:10:23 : Data read successfully
22:10:23 : Time elapsed during the read operation is: 00:00:00.006
22:10:26 : UPLOADING -
22:10:26 : Size : 1024 Bytes
22:10:26 : Address : 0x08000000
22:10:26 : Read progress:
22:10:26 : Data read successfully
22:10:26 : Time elapsed during the read operation is: 00:00:00.006
22:10:33 : UPLOADING -
22:10:33 : Size : 1024 Bytes
22:10:33 : Address : 0x0800e000
22:10:33 : Read progress:
22:10:33 : Data read successfully
22:10:33 : Time elapsed during the read operation is: 00:00:00.006
22:10:49 : UPLOADING -
22:10:49 : Size : 1024 Bytes
22:10:49 : Address : 0x0801e000
22:10:49 : Read progress:
22:10:49 : Data read successfully
22:10:49 : Time elapsed during the read operation is: 00:00:00.006

100%

Apply the modifications and reset the board.

Step 5 : Check the secure memory

During this step, you have to disconnect and reconnect but with HOTPLUG mode. Once it is realized you should not be able to see 0x08000000.

The screenshot shows the STM32CubeProgrammer interface. The 'Memory & File edition' window is active, displaying the 'Device memory' tab. The address is set to 0x08000000, size is 0x400, and data width is 32-bit. A 'Read' button is visible. An error dialog box is displayed in the center, stating 'Error: Data read failed'. The log window at the bottom shows the following messages:

```

22:35:21: UPLOADING OPTION BYTES DATA ...
22:35:21: Bank : 0x00
22:35:21: Address : 0x40020000
22:35:21: Size : 112 Bytes
22:35:21: UPLOADING ...
22:35:21: Size : 1024 Bytes
22:35:21: Address : 0x08000000
22:35:21: Read progress
22:35:21: Error: Data read failed
22:35:21: UPLOADING ...
22:35:21: Size : 1024 Bytes
22:35:21: Address : 0x081E0000
22:35:21: Read progress
22:35:21: Data read successfully
22:35:21: Time elapsed during the read operation is 00:00:00.006
22:36:47: UPLOADING ...
22:36:47: Size : 1024 Bytes
22:36:47: Address : 0x08000000
22:36:47: Read progress
22:36:47: Error: Data read failed
  
```

But at the same time you should be able to see the 0x0801E000 memory section that corresponds to the unsecured function.

The screenshot shows the STM32CubeProgrammer interface. The 'Memory & File edition' window is active, displaying the 'Device memory' tab. The address is set to 0x0801E000, size is 0x400, and data width is 32-bit. The 'Read' button is visible. The memory map is displayed, showing the 0x0801E000 section highlighted. The log window at the bottom shows the following messages:

```

22:35:21: CONNECTING TO THE HOT PLUG MODE ...
22:35:21: Connect mode Hot Plug
22:35:21: Reset mode : Hardware reset
22:35:21: Device ID : 0x480
22:35:21: Revision ID : Rev B
22:35:21: UPLOADING OPTION BYTES DATA ...
22:35:21: Bank : 0x00
22:35:21: Address : 0x40020000
22:35:21: Size : 112 Bytes
22:35:21: UPLOADING ...
22:35:21: Size : 1024 Bytes
22:35:21: Address : 0x08000000
22:35:21: Read progress
22:35:21: Error: Data read failed
22:35:21: UPLOADING ...
22:35:21: Size : 1024 Bytes
22:35:21: Address : 0x081E0000
22:35:21: Read progress
22:35:21: Data read successfully
22:35:21: Time elapsed during the read operation is 00:00:00.006
  
```