

ACTIVATE SECURE MEMORY

Link

<https://www.youtube.com/watch?v=qCuVBD2dmTA&list=PLnMKNibPkDnFzux3PHKUEi14ftDn9Cbm7&index=13>

Description

In this paperwork, we will configure the secure an application loaded in the memory, and then secure portion 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 A blinky code	4
Step 3 : Separate the memory	5
Step 4 : Now create a bootloader code.	6
Step 5 : Secure the memory.....	7

Prerequisites

Security Features by STM32 Series

7

STM32 Series	Security Features																
	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	Arm Cortex®
STM32 F0	Available on all devices	Available on all devices		Available on all devices							Available on all devices					Available on all devices	M0
STM32 F1	Available on all devices	Available on all devices		Available on all devices			Available on all devices				Available on all devices					Available on all devices	M3
STM32 F2	Available on all devices	Available on all devices		Available on all devices			Available on all devices				Available on all devices	Depends on device part number	Depends on device part number	Depends on device part number		Available on all devices	M3
STM32 F3	Available on all devices	Available on all devices		Available on all devices			Available on all devices				Available on all devices					Available on all devices	M4
STM32 F4	Available on all devices	Available on all devices	Depends on device part number	Available on all devices			Available on all devices				Available on all devices	Depends on device part number	Depends on device part number	Depends on device part number		Available on all devices	M4
STM32 F7	Available on all devices	Available on all devices	Depends on device part number	Available on all devices			Available on all devices				Available on all devices	Depends on device part number	Depends on device part number	Depends on device part number		Available on all devices	M7
STM32 L0	Available on all devices	Available on all devices		Available on all devices			Available on all devices	Available on all devices			Available on all devices		Depends on device part number			Available on all devices	M0+
STM32 L1	Available on all devices	Available on all devices		Available on all devices			Available on all devices				Available on all devices					Available on all devices	M3
STM32 L4	Available on all devices	Available on all devices		Available on all devices			Available on all devices	Available on all devices			Available on all devices		Depends on device part number	Depends on device part number		Available on all devices	M4
STM32 L5	Available on all devices	Available on all devices		Available on all devices			Available on all devices		Available on all devices	Available on all devices	Available on all devices				Available on all devices	Available on all devices	M33
STM32 H7	Available on all devices	Available on all devices		Available on all devices	Depends on device part number	Depends on device part number	Available on all devices				Available on all devices	Depends on device part number	Depends on device part number	Depends on device part number		Available on all devices	M7/M4
STM32 G0	Available on all devices	Available on all devices		Available on all devices		Depends on device part number	Available on all devices				Available on all devices	Depends on device part number	Depends on device part number			Available on all devices	M0+
STM32 G4	Available on all devices	Available on all devices		Available on all devices		Depends on device part number	Available on all devices				Available on all devices	Depends on device part number	Depends on device part number			Available on all devices	M4
STM32 WB	Available on all devices	Available on all devices		Available on all devices			Available on all devices				Available on all devices				Available on all devices	Available on all devices	M4/M0+

Available on all devices

Depends on device part number

life.augmented

47

STM32 Board

ST-Link cable

STM32CubeProgrammer

STM32CubeMX

STM32CubeIDE

Walkthrough

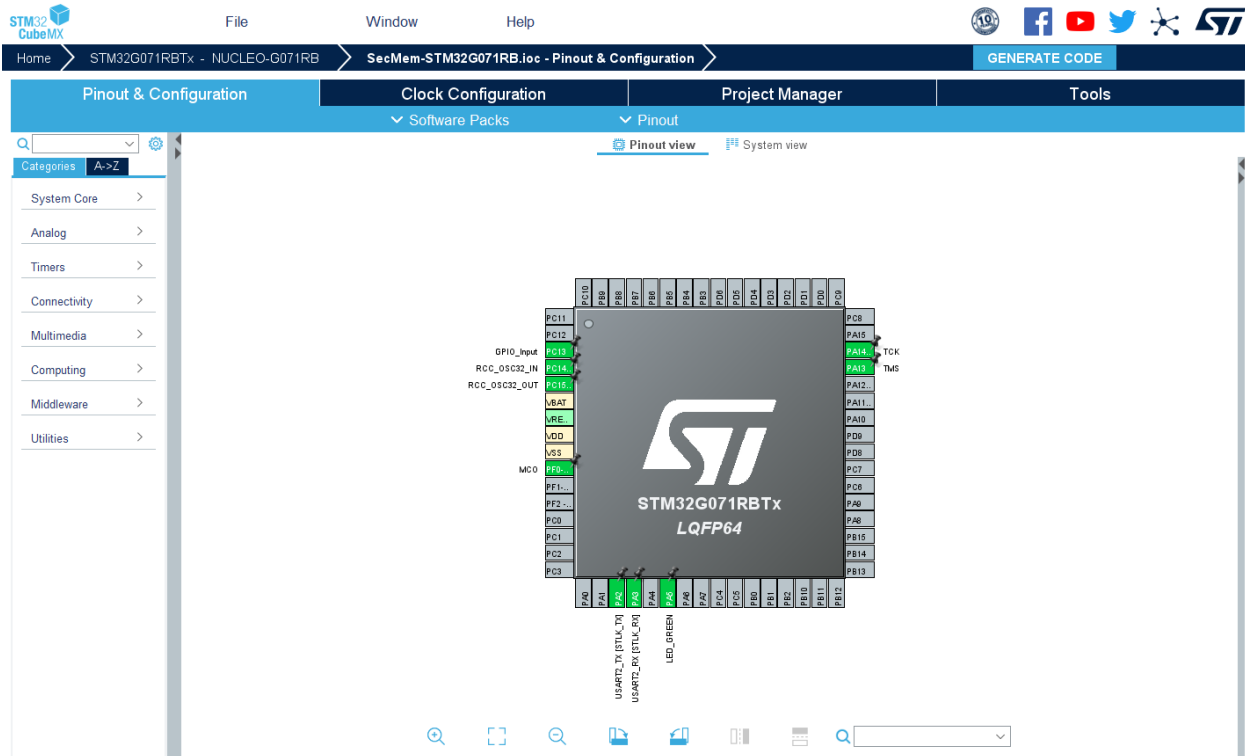
Step 1 : Launch STM32CubeMX and generate the code

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.

Generate to exact same projects.



Step 2 : Create A blinky code

To do this blinky code, we will set the memory vector in our code. (It is not the same as in the video because I guess that now those functions take more memory and can't compile).

This will put our code in this section of memory.

```
int main(void)
{
    /* USER CODE BEGIN 1 */
    SCB->VTOR=0x800E000;
    /* USER CODE END 1 */

    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_USART2_UART_Init();
    /* USER CODE BEGIN 2 */

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        /* USER CODE END WHILE */

        /* USER CODE BEGIN 3 */
        HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin);
        HAL_Delay(500);
        /* USER CODE END 3 */
    }
}
```

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.

```

16 **
17 ** Target      : STMicroelectronics STM32
18 **
19 ** Distribution: The file is distributed as is, without any warranty
20 **              of any kind.
21 **
22 ****
23 ** @attention
24 **
25 ** <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
26 ** All rights reserved.</center></h2>
27 **
28 ** This software component is licensed by ST under BSD 3-Clause license,
29 ** the "License"; You may not use this file except in compliance with the
30 ** License. You may obtain a copy of the License at:
31 **      opensource.org/licenses/BSD-3-Clause
32 **
33 ****
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 = 0x800E000,  LENGTH = 16K
50 }
51
52 /* Sections */
53 SECTIONS
54 {
55   /* The startup code into "FLASH" Rom type memory */
56   .isr_vector :
57   {
58     . = ALIGN(4);
59     KEEP(*(.isr_vector)) /* Startup code */
60     . = ALIGN(4);
61   } >FLASH
62 }

```

When you have finished you can compile the code and execute the debugger. You will see that the light is blinking.

Step 4 : Now create a bootloader code.

The following code will load the code located at 0x800E00 (our blinky application) when the user button is pressed.

```
/* USER CODE END PV */

/* Private function prototypes -----*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code -----*/
/* USER CODE BEGIN 0 */
#define BL_EXIT_STICKY 0x1FFF6800
#define MAGIC_NUMBER 0x08192A3C
#define APPLICATION_ADDRESS 0x800E000

typedef void (*pFunction)(uint32_t a, uint32_t b, uint32_t c);

pFunction JumpToApplication;
uint32_t JumpAddress;

/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash interface and the SysTick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_USART2_UART_Init();
    /* USER CODE BEGIN 2 */

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        /* USER CODE END WHILE */

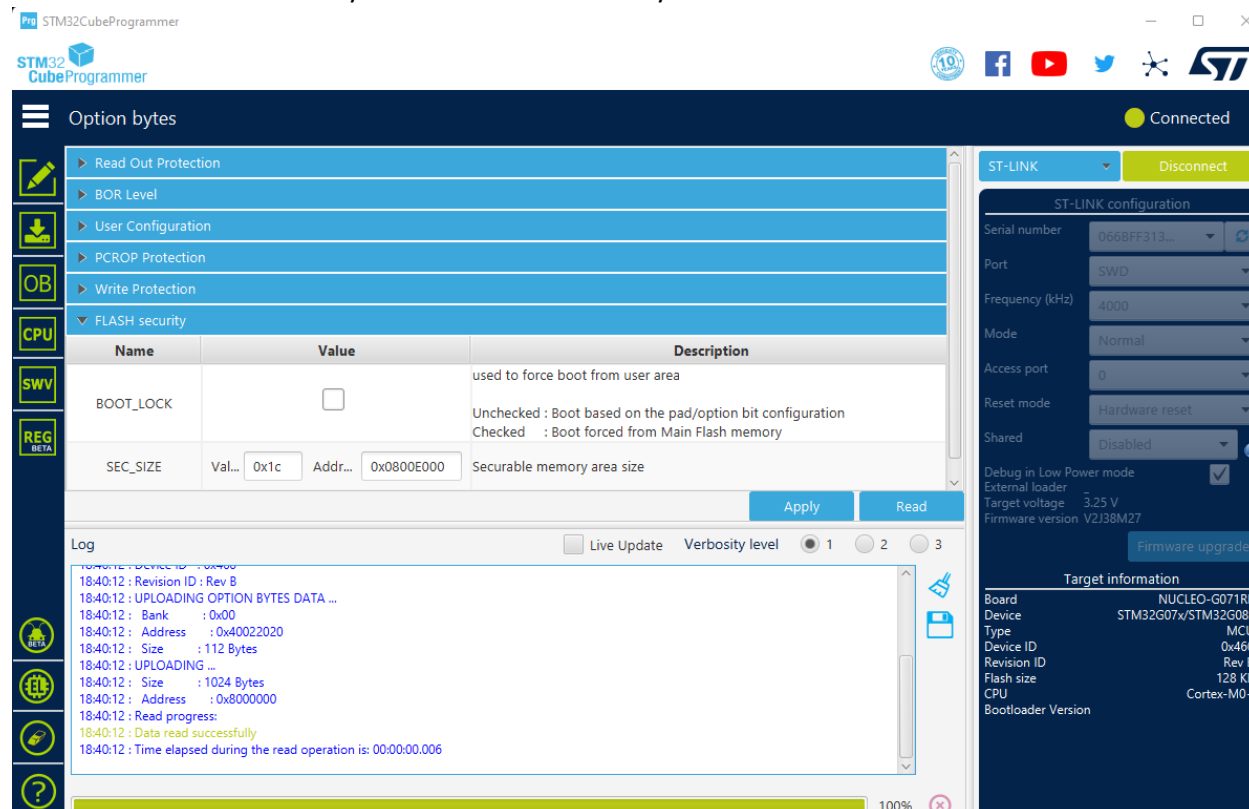
        /* USER CODE BEGIN 3 */
        if ( HAL_GPIO_ReadPin(BUTTON_BLUE_GPIO_Port, BUTTON_BLUE_Pin) == 0)
        {
            JumpAddress= *((__IO uint32_t*)(BL_EXIT_STICKY+4));
            JumpToApplication = (pFunction)JumpAddress;
            JumpToApplication(JumpAddress, MAGIC_NUMBER, APPLICATION_ADDRESS);
        }
    }
    /* USER CODE END 3 */
}
```

When you have finished you can compile the code and execute the debugger. When you will push the button, the led will start to blink.

Step 5 : Secure the memory

To secure the memory, open CubeProgrammer, go to OB and in the FLASH Security set the address of the memory location of your blinky application as the unsecure part.

You couldn't read at memory location 8x8000000 but you could see the code at 0x800E000.



The screenshot shows the STM32CubeProgrammer application window. The 'Option bytes' tab is selected, and the 'FLASH security' section is expanded. The 'SEC_SIZE' field is set to 'Val... 0x1c' and 'Addr... 0x0800E000'. The 'BOOT_LOCK' checkbox is unchecked. The 'Log' window at the bottom shows the following messages:

```

18:40:12 : Revision ID : Rev B
18:40:12 : UPLOADING OPTION BYTES DATA ...
18:40:12 : Bank : 0x00
18:40:12 : Address : 0x40022020
18:40:12 : Size : 112 Bytes
18:40:12 : UPLOADING ...
18:40:12 : Size : 1024 Bytes
18:40:12 : Address : 0x8000000
18:40:12 : Read progress:
18:40:12 : Data read successfully
18:40:12 : Time elapsed during the read operation is: 00:00:00.006
  
```

The 'Target information' panel on the right shows the following details:

- Board: NUCLEO-G071RB
- Device: STM32G07x/STM32G08x
- Type: MCU
- Device ID: 0x460
- Revision ID: Rev B
- Flash size: 128 KB
- CPU: Cortex-M0+
- Bootloader Version: -