

Tampering Protection

Link

<https://www.youtube.com/watch?v=qCuVBD2dmTA&list=PLnMKNibPkDnFzux3PHKUEi14ftDn9Cbm7&index=21>


Description

In this paperwork, we will learn how to setup and configure a firewall on our STM32 board.

Contents

Link.....	1
Description.....	1
Prerequisites	2
STM32 Board.....	2
ST-Link cable	2
STM32CubeMX	2
STM32CubeIDE.....	2
Walkthrough	3
Step 1 : Launch STM32CubeMX and generate the code	3
Step 2 : Write the main	4
Step 3 : Check the protection	5

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	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		Depends on device part number	Depends on device part number		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	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		Depends on device part number	Depends on device part number		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	Depends on device part number		Available on all devices	M4
STM32 L5	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	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	Available on all devices	Depends on device part number	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		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	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	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		Depends on device part number		Available on all devices	Available on all devices	M4/M0+

Available on all devices

Depends on device part number


life.augmented

STM32 Board

ST-Link cable

STM32CubeMX

STM32CubeIDE

Walkthrough

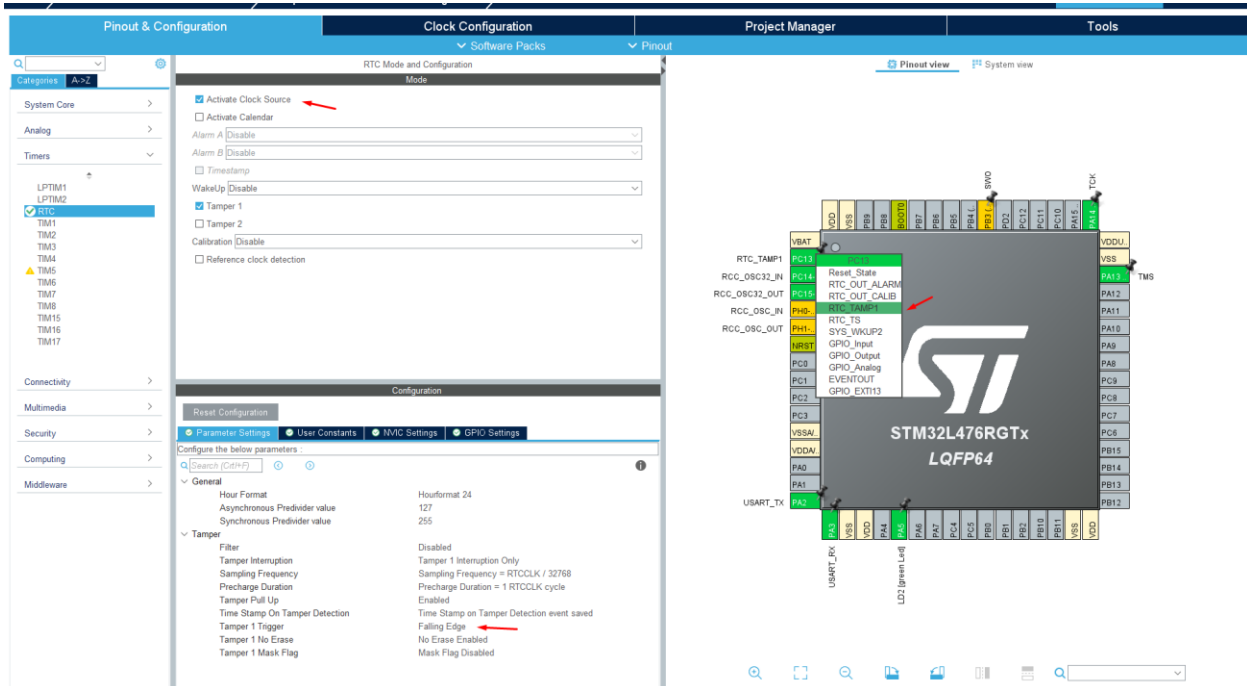
Step 1 : Launch STM32CubeMX and generate the code

Launch STM32CubeMX and select the right board depending on the one you are using. In my case I use the L476RG Nucleo board. Then you can generate the code of your project.

Then you have to replace the button with RTC_TAMP1.

Then you will have to go in the RTC process and activate the clock source. In the Parameters settings, go to Tamper 1 Trigger and activate Falling Edge.

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



Step 2 : Write the main

Then we will rewrite the code for the Tamper callback. In this one we will toggle the LED to be sure that we activated the function once the button is pushed.

In the main, we will put data in two backup registries.

```
/* USER CODE BEGIN 0 */
void HAL_RTCEx_Tamper1EventCallback(RTC_HandleTypeDef *hrtc)
{
    HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
}
/* 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 */

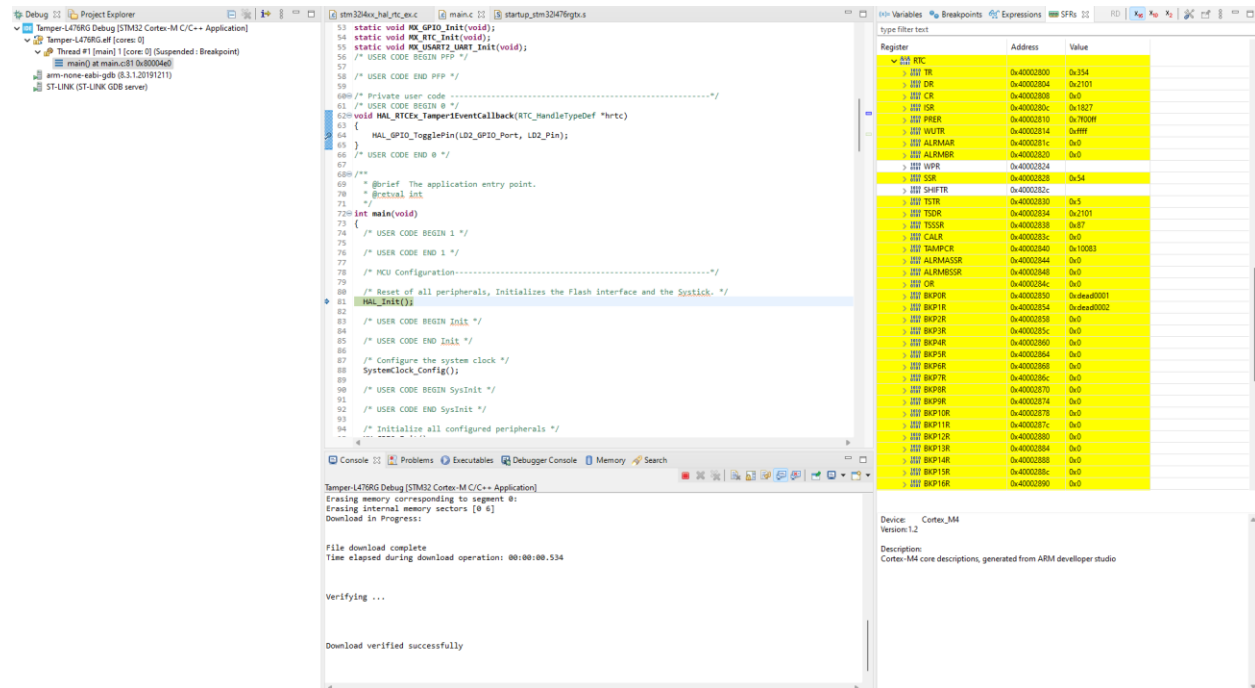
    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_RTC_Init();
    MX_USART2_UART_Init();
    /* USER CODE BEGIN 2 */
    HAL_RTCEx_BKUPWrite(&hrtc, RTC_BKP_DR0, 0xdead0001);
    HAL_RTCEx_BKUPWrite(&hrtc, RTC_BKP_DR1, 0xdead0002);
    /* USER CODE END 2 */
}
```

If the code is completed, just compile and launch the debug.

Step 3 : Check the protection

Just launch the code and go to the register section.

You can see in the register values that our two values are in the memory.



The screenshot shows the STM32CubeIDE IDE with the main.c file open. The code is for a STM32L476RG microcontroller. The Register window on the right shows the values of various registers. The Register window is filtered by 'RTC'. The registers shown are:

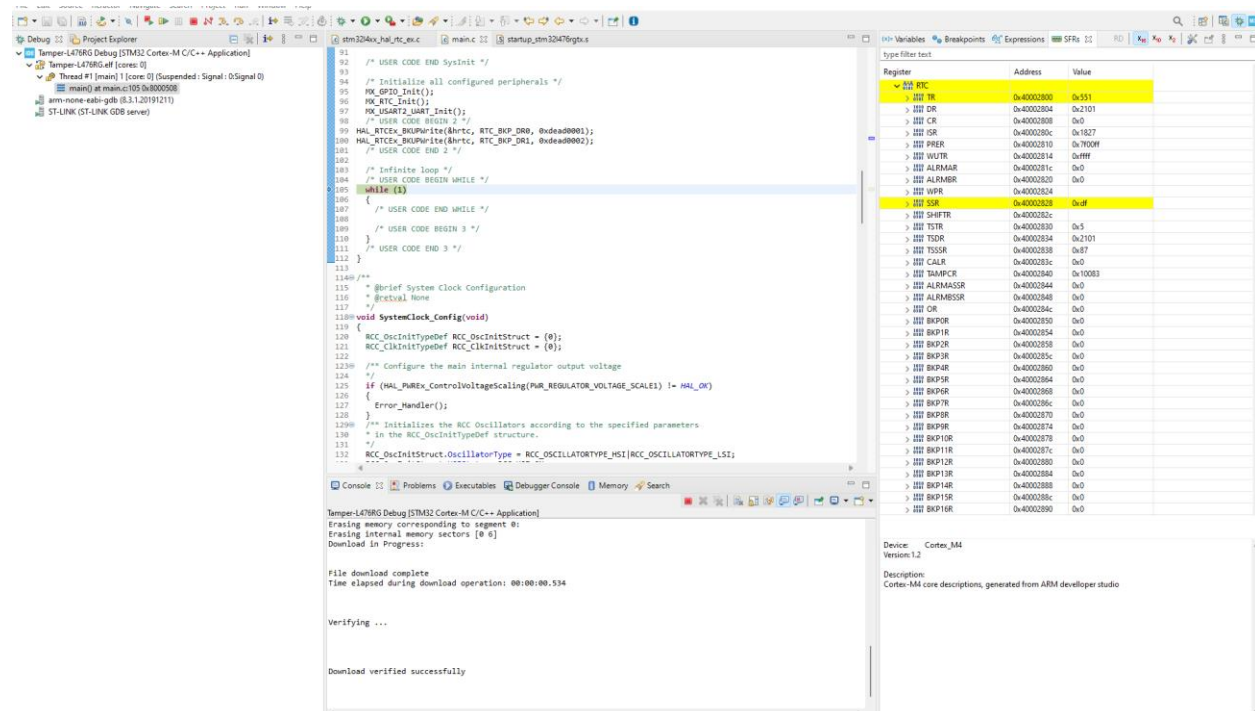
Register	Address	Value
RTC_CR	0x40020000	0x134
RTC_DR	0x40020004	0x2101
RTC_TR	0x40020008	0x0
RTC_OR	0x4002000C	0x1827
RTC_PRER	0x40020010	0x7000F
RTC_WUTR	0x40020014	0xFFFF
RTC_ALRMAAR	0x40020018	0x0
RTC_ALRMMR	0x4002001C	0x0
RTC_WPR	0x40020024	0x0
RTC_SSR	0x40020028	0x54
SHIFTR	0x4002002C	0x0
TSR	0x40020030	0x5
TSOR	0x40020034	0x2101
TSSR	0x40020038	0x0
CALR	0x4002003C	0x0
TAMPCFR	0x40020040	0x10003
ALRAMISR	0x40020044	0x0
ALRAMISR	0x40020048	0x0
OR	0x4002004C	0x0
BKPR	0x40020050	0x0
BKPR	0x40020054	0x0
BKPR	0x40020058	0x0
BKPR	0x4002005C	0x0
BKPR	0x40020060	0x0
BKPR	0x40020064	0x0
BKPR	0x40020068	0x0
BKPR	0x4002006C	0x0
BKPR	0x40020070	0x0
BKPR	0x40020074	0x0
BKPR	0x40020078	0x0
BKPR	0x4002007C	0x0
BKPR	0x40020080	0x0
BKPR	0x40020084	0x0
BKPR	0x40020088	0x0
BKPR	0x4002008C	0x0
BKPR	0x40020090	0x0

The Console window shows the following output:

```

Tamper-L476RG Debug [STM32 Cortex-M C/C++ Application]
Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 6]
Download in Progress:
File download complete
Time elapsed during download operation: 00:00:00.534
Verifying ...
Download verified successfully
  
```

Once the user button is pushed (simulating physical tampering), you can see that the data has been erased :



The screenshot shows the STM32CubeIDE IDE with the main.c file open. The code is for a STM32L476RG microcontroller. The Register window on the right shows the values of various registers. The Register window is filtered by 'RTC'. The registers shown are:

Register	Address	Value
RTC_CR	0x40020000	0x131
RTC_DR	0x40020004	0x2101
RTC_TR	0x40020008	0x0
RTC_OR	0x4002000C	0x1827
RTC_PRER	0x40020010	0x7000F
RTC_WUTR	0x40020014	0xFFFF
RTC_ALRMAAR	0x40020018	0x0
RTC_ALRMMR	0x4002001C	0x0
RTC_WPR	0x40020024	0x0
RTC_SSR	0x40020028	0xFF
SHIFTR	0x4002002C	0x0
TSR	0x40020030	0x5
TSOR	0x40020034	0x2101
TSSR	0x40020038	0x0
CALR	0x4002003C	0x0
TAMPCFR	0x40020040	0x10003
ALRAMISR	0x40020044	0x0
ALRAMISR	0x40020048	0x0
OR	0x4002004C	0x0
BKPOR	0x40020050	0x0
BKPR	0x40020054	0x0
BKPR	0x40020058	0x0
BKPR	0x4002005C	0x0
BKPR	0x40020060	0x0
BKPR	0x40020064	0x0
BKPR	0x40020068	0x0
BKPR	0x4002006C	0x0
BKPR	0x40020070	0x0
BKPR	0x40020074	0x0
BKPR	0x40020078	0x0
BKPR	0x4002007C	0x0
BKPR	0x40020080	0x0
BKPR	0x40020084	0x0
BKPR	0x40020088	0x0
BKPR	0x4002008C	0x0
BKPR	0x40020090	0x0

The Console window shows the following output:

```

Tamper-L476RG Debug [STM32 Cortex-M C/C++ Application]
Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 6]
Download in Progress:
File download complete
Time elapsed during download operation: 00:00:00.534
Verifying ...
Download verified successfully
  
```