Tampering Protection

Link

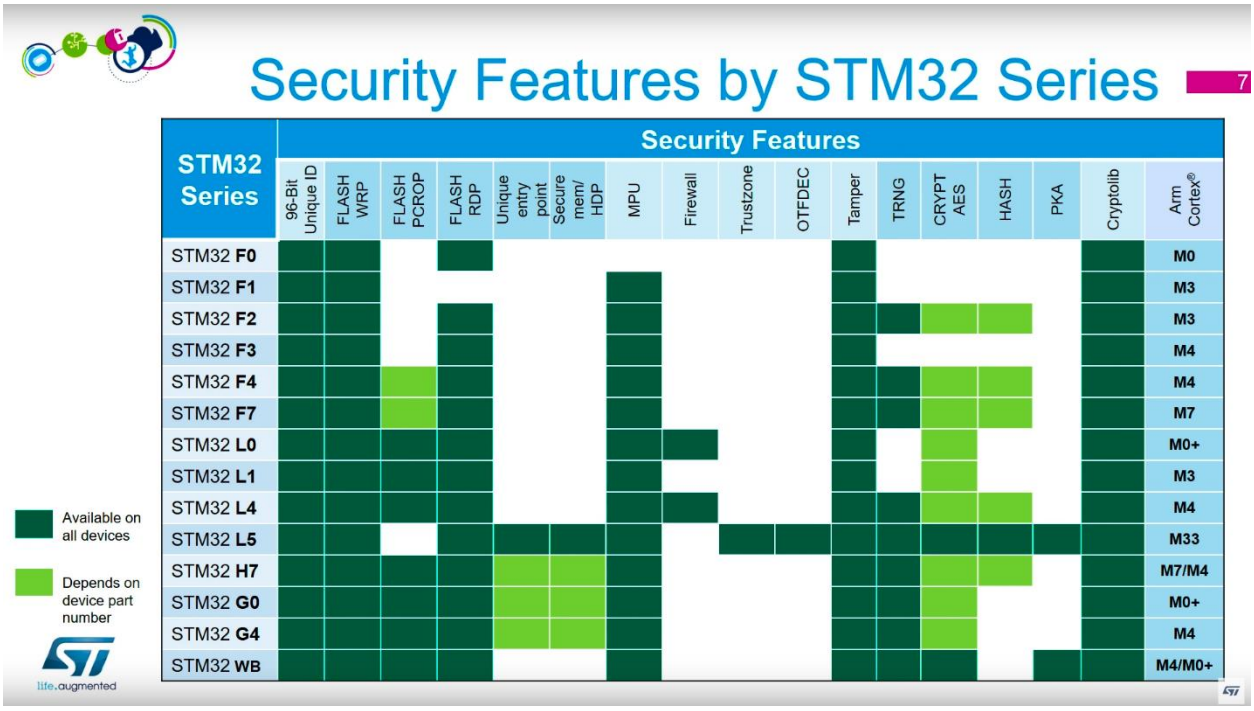
https://www.youtube.com/watch?v=qCuVBD2dmTA&list=PLnMKNibPkDnFzux3PHKUEi14ftDn9Cbm7&index=21

Description

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

Contents

Prerequisites



STM32 Board

ST-Link cable

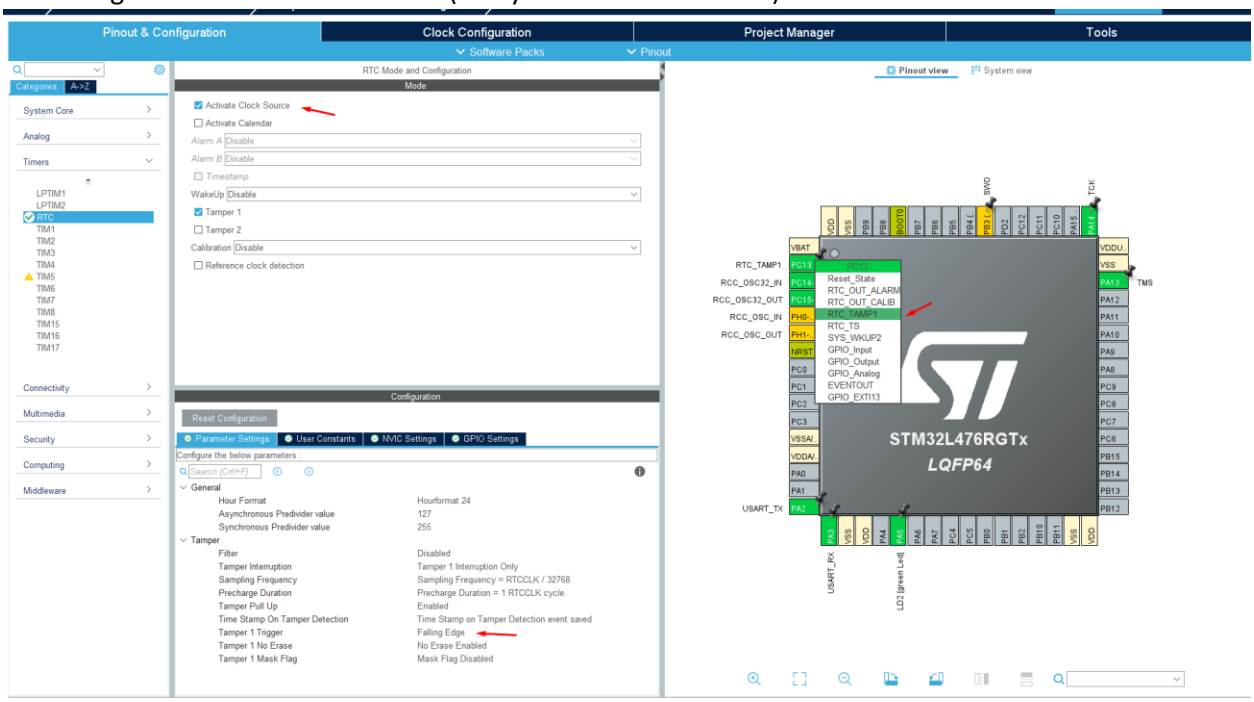
STM32CubeMX

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.

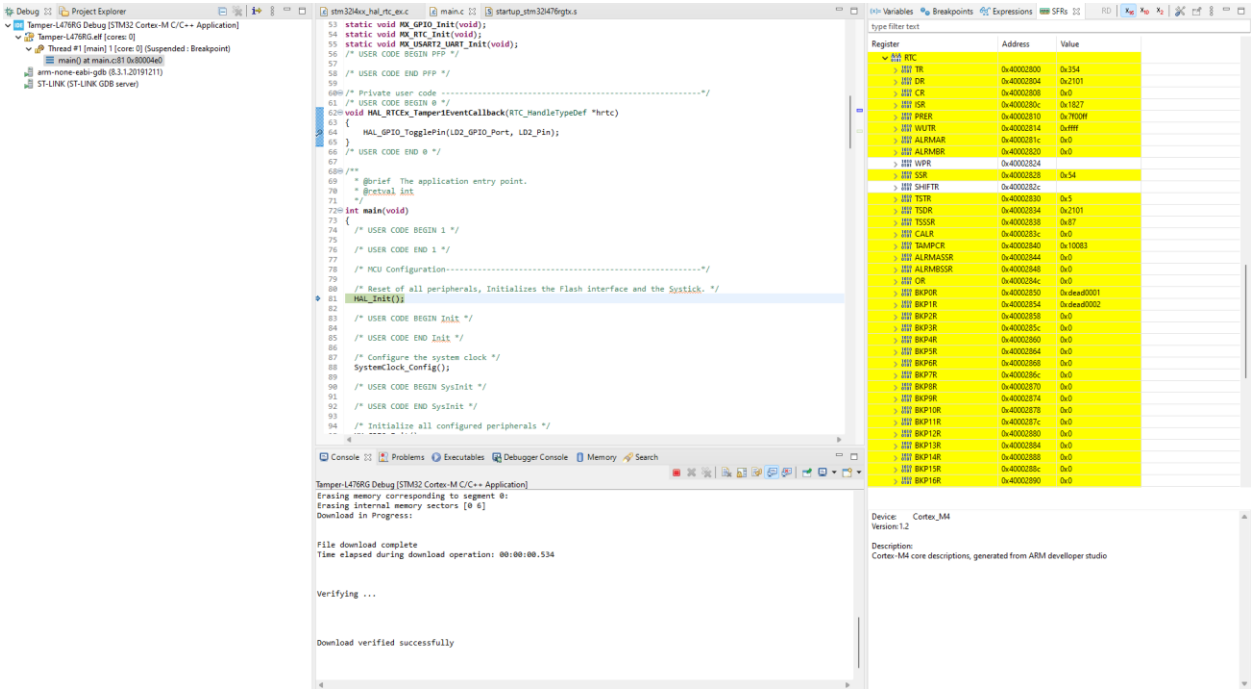
```
/* USER CODE BEGIN 0 */
void HAL_RTCEx_Tamper1EventCallback(RTC_HandleTypeDef *hrtc)
{
    HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
}
/* USER CODE END 0 */

/**
  * @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.



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