STM32 Tamper Protection

This tutorial will demonstrate the tamper protection feature.

- Purpose: detect physical tampering in a secure application and to automatically erase sensitive data in case of intrusion.
- On detection:
  - Erase of backup register
  - Prohibit access to the backup SRAM or erase it(STM32L5)
  - Can generate a timestamp event
- Available on family

If an attack shorts the tamper input pin to the inactive state, then there is no tamper detection.

Hardware:

- Nucleo-L476RG board(64-pin),available at: www.st.com/en/evaluation-tools/nucleo-l476rg.html
- Standard-A -to- Mini USB cable

Literature:

- STM32L476xx Datasheet
- UM1724 User manual STM32 Nucleo-64 boards
- UM1884 Description of STM32L4/L4+ HAL and low-layer drivers
- UM1718 User manual STM32CubeMX for STM32 configuration and initialization C code generation
- RM0351 Reference Manual

Stages

- 1: Create Code with CubeIDE
- 2: Simulate Tamper event with Pushbutton

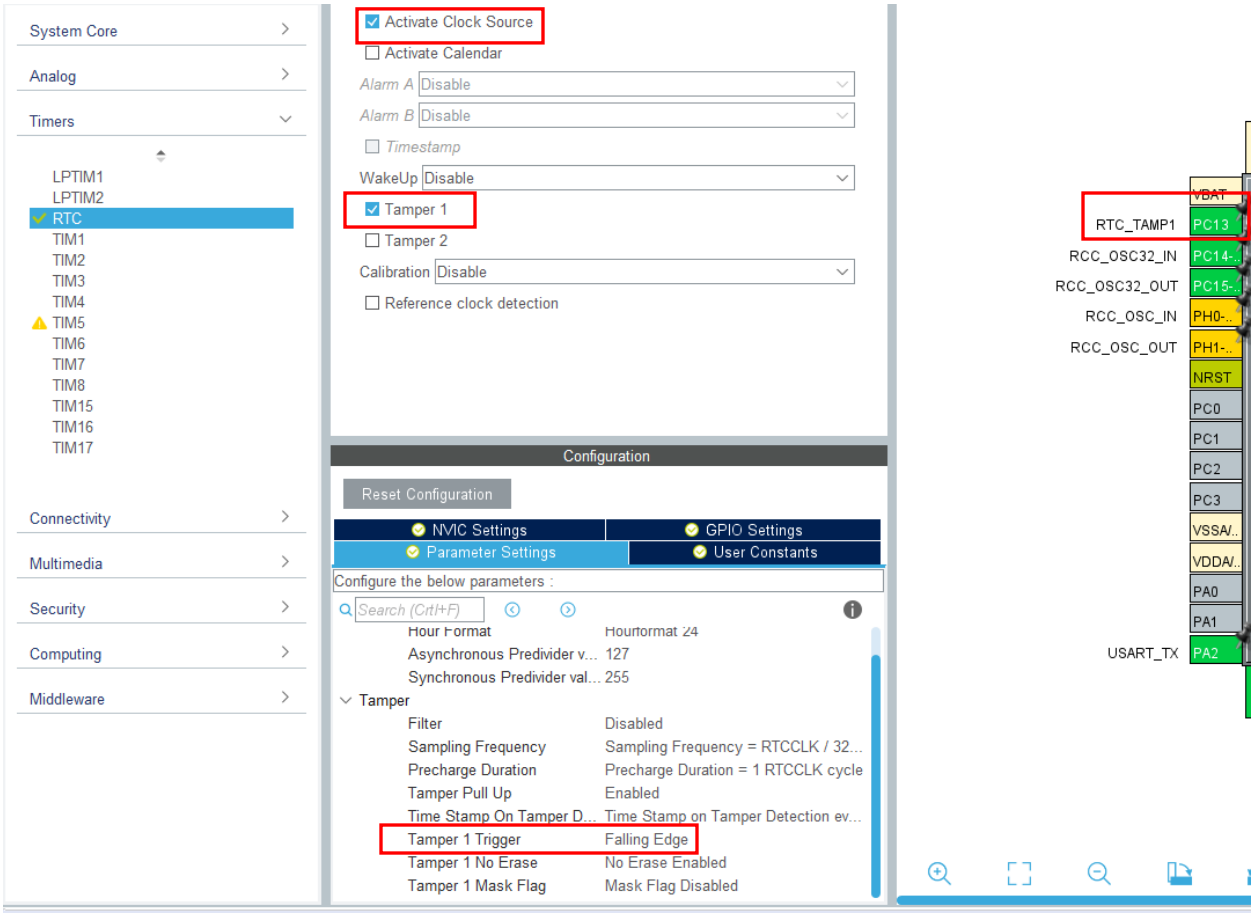
1: Create Code with CubeIDE

First we create a STM32 project. Open STM32CubeIDE. Go to *File->New->STM32 Project.* Go to board selector and select Nucleo-L476RG. When prompted to initialize peripherals in their default mode, select yes.

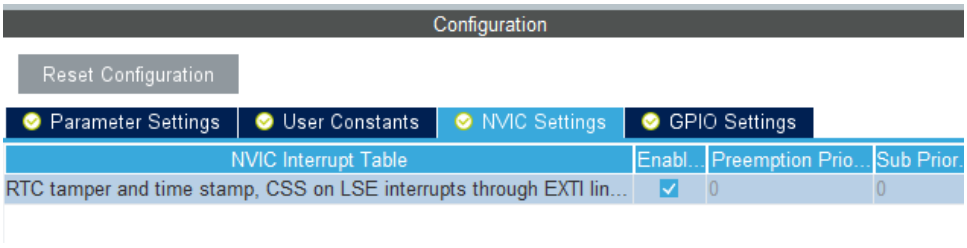
In the .ioc file, go to *Timers->RTC.* We need to change PC13 to not be a GPIO_EXTI13. In the Pinout diagram, click on PC13 and select RTC_TAMP1.

Next, check the box labeled “Activate Clock Source” and also check the box labeled “Tamper 1.” Next under Parameter Settings, go to “Tamper 1 Trigger” and change it to Falling Edge.

This will configure the Pushbutton to simulate a tamper event.



Next, go to NVIC Settings and check “RTC tamper time stamp, CSS on LSE interrupts through EXTI line 19.”



Save the project and select yes when asked to generate code.

Open the interrupt file and we can see the interrupt handler will call a IRQHandler function.

Under Core->Src open stm32l4xx_it.c. We can see:

```
203 void TAMP_STAMP_IRQHandler(void)
204 {
205   /* USER CODE BEGIN TAMP_STAMP_IRQn 0 */
206
207   /* USER CODE END TAMP_STAMP_IRQn 0 */
208   HAL_RTCEx_TamperTimeStampIRQHandler(&hrtc);
209   /* USER CODE BEGIN TAMP_STAMP_IRQn 1 */
210
211   /* USER CODE END TAMP_STAMP_IRQn 1 */
212 }
213
```

We want to edit a Callback function. In the project explorer go to Drivers-> STM32L4xx_HAL_Driver ->Src and open stm32l4xx_hal_rtc_ex.c. Press control+F and search for "__weak." Keep navigating until the function shown in the image below is found:

```
__weak void HAL_RTCEx_Tamper1EventCallback(RTC_HandleTypeDef *hrtc)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(hrtc);

  /* NOTE : This function should not be modified, when the callback is needed,
            the HAL_RTCEx_Tamper1EventCallback could be implemented in the user file
   */
}
```

Copy the function prototype to main.c and edit as shown below:

```
60 /* Private user code ---------------------------------------------------------*/
61 /* USER CODE BEGIN 0 */
62 void HAL_RTCEx_Tamper1EventCallback(RTC_HandleTypeDef *hrtc){
63     HAL_GPIO_TogglePin(LD2_GPIO_Port,LD2_Pin);
64 }
65 /* USER CODE END 0 */
66
```

Build and Debug the project.

2: Simulate Tamper event with Pushbutton

Resume the execution and push the blue button. We can see the LED toggle.

Now let us put data into the backup registers and see that it will erase on the tamper event. Add the lines of code in int main() of main.c.

```
 97   /* USER CODE BEGIN 2 */
 98   HAL_RTCEx_BKUPWrite(&hrtc, RTC_BKP_DR0,0xdead0001);
 99   HAL_RTCEx_BKUPWrite(&hrtc, RTC_BKP_DR1,0xdead0002);
100   /* USER CODE END 2 */
101
102   /* Infinite loop */
103   /* USER CODE BEGIN WHILE */
104   while (1)
105   {
```

Build and Debug Project. Go to Window->Show View ->SFRs. In the SFRs window, expand RTC. Step through the code to see the backup registers with the data we wrote in them.

```
 97   /* USER CODE BEGIN 2 */
 98   HAL_RTCEx_BKUPWrite(&hrtc, RTC_BKP_DR0,0xdead0001);
 99   HAL_RTCEx_BKUPWrite(&hrtc, RTC_BKP_DR1,0xdead0002);
100   /* USER CODE END 2 */
101
102   /* Infinite loop */
103   /* USER CODE BEGIN WHILE */
104   while (1)
105   {
106     /* USER CODE END WHILE */
107
```

ALRMBSSR	0x40002848	0x0
OR	0x4000284c	0x0
BKP0R	0x40002850	0xdead0001
BKP1R	0x40002854	0xdead0002
BKP2R	0x40002858	0x0
BKP3R	0x4000285c	0x0
BKP4R	0x40002860	0x0
BKP5R	0x40002864	0x0
BKP6R	0x40002868	0x0

Next add a breakpoint to the tamper IRQ handler function in the interrupt file(stm32l4xx_it.c).

```
203 void TAMP_STAMP_IRQHandler(void)
204 {
205   /* USER CODE BEGIN TAMP_STAMP_IRQn 0 */
206
207   /* USER CODE END TAMP_STAMP_IRQn 0 */
208   HAL_RTCEx_TamperTimeStampIRQHandler(&hrtc);
209   /* USER CODE BEGIN TAMP_STAMP_IRQn 1 */
210
211   /* USER CODE END TAMP_STAMP_IRQn 1 */
212 }
213
```

Resume the execution and press the blue button. You will the backup registers have been erased.