ACTIVATE WRP

Link

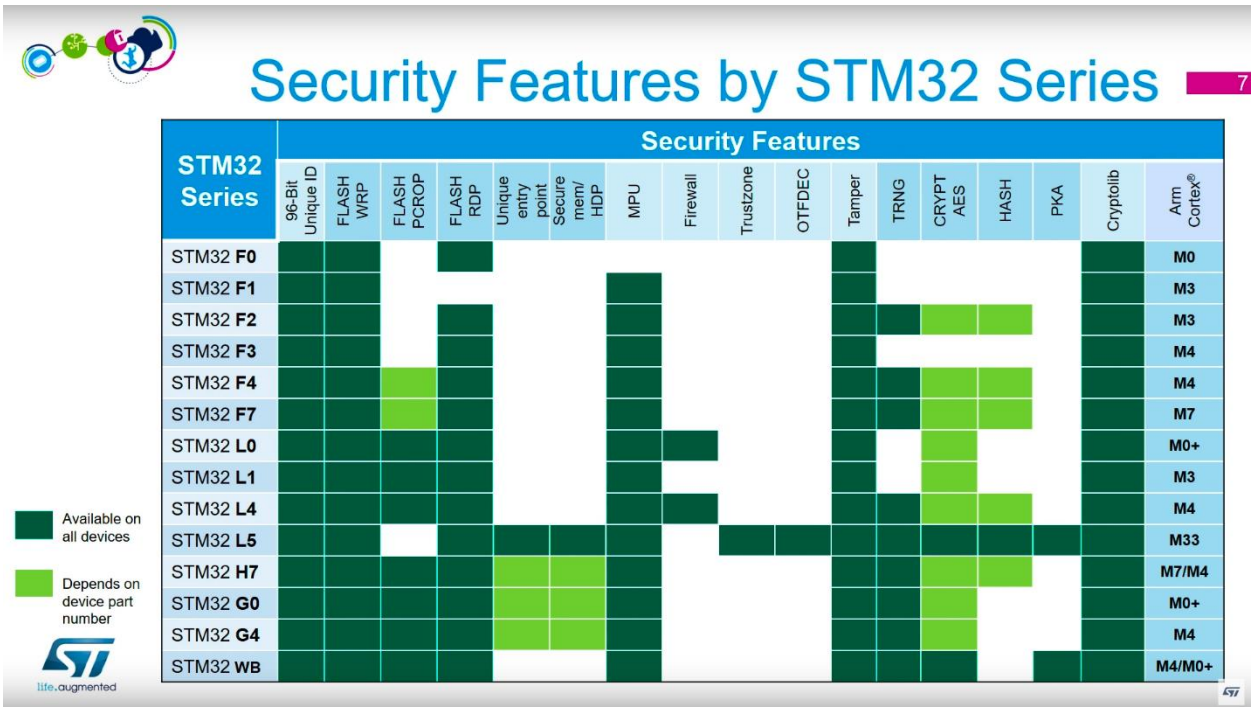
https://www.youtube.com/watch?v=qCuVBD2dmTA&list=PLnMKNibPkDnFzux3PHKUEi14ftDn9Cbm7&index=5

Description

In this paperwork, we will do all the steps to quickly activate the write protection (WRP) of the stm32 board we are using.

Contents

Prerequisites



STM32 Board

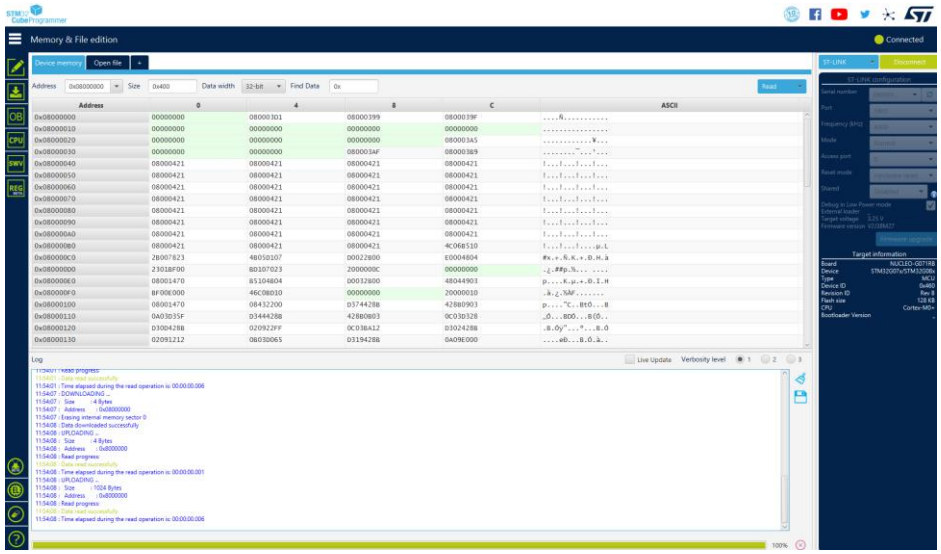
ST-Link cable

STM32CubeProgrammer

Walkthrough

Step 1 : Run STM32CubeProgrammer

Launch STM32CubeProgrammer. On it you will just connect with your STM32 board, using your ST-Link. Once connected you can check that you actually can change data by double-clicking and changing value.

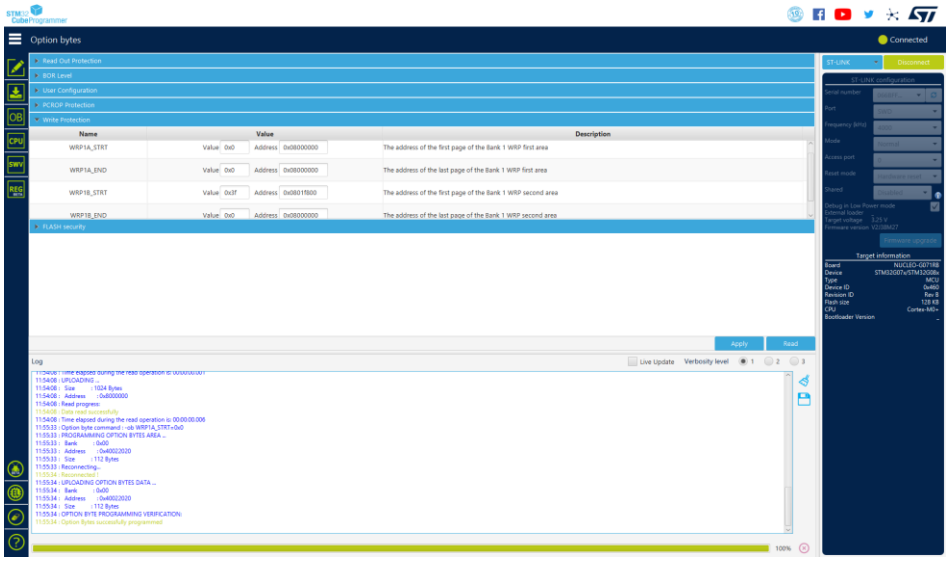


Step 2 : Activate Write Protection

Once you've tested that you can change values, you go to the OB menu and click on the write protection section. To activate a protection set the following :

WP1IA_STRT : Value : 0x0 (this is the first address being protected)

WP1IA_STRT : Value : 0x0 (this is the last address being protected)



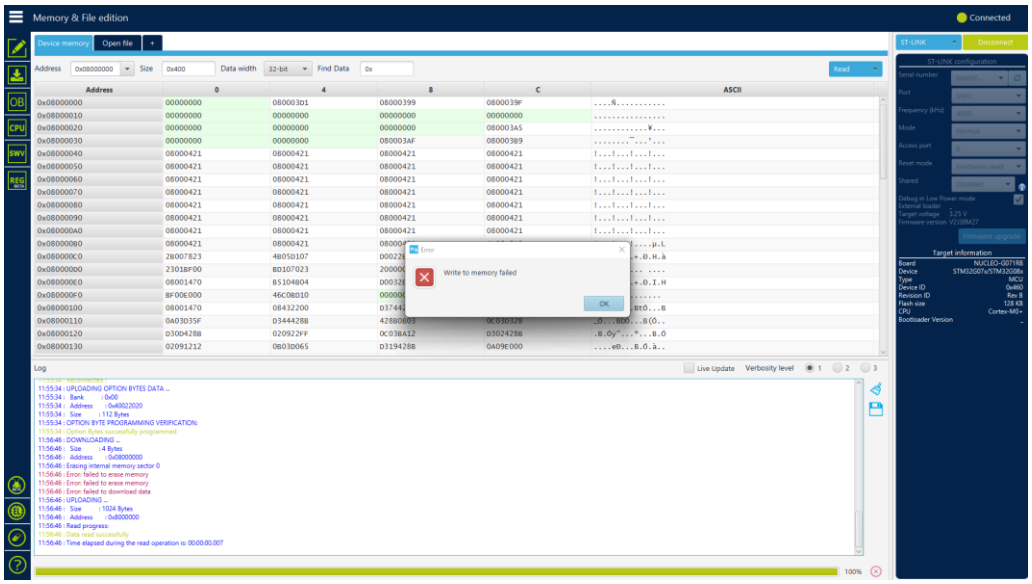
Step 3 : Test the writing protection

Come back to device memory and now you should not be able to modify the first address (on the top left corner). If you do so, you will get an error message confirming us that the write protection has been activated.

For disabling it again put the go back to the previous step and set those values :

WP1IA_STRT : Value : 0x0 (this is the first address being protected)

WP1IA_STRT : Value : 0x0 (this is the last address being protected)



Step 4 : Develop the protection activation

To do so, you will have to generate the code for the L476RG nucleo board. It doesn't work on the G0 and the WB55 that are furnished.

Once the code is generated, just add the code pointed with a red arrow in the main file.

The code basically enter in write protection mode when the button is pushed.

```
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(B1_GPIO_Port, B1_Pin) == 0)
          {
              SYSCFG->SWPR = 0x1;
          }
  }
  /* USER CODE END 3 */
}
```

Step 5 : Edit the memory content

To do so you will need to use the tool to access the memory on the IDE. Once you are on It, you can simply go on address 0x10000000.

Put the code in pause and modify the memory. No error appears and you can continue the program

STM32G071RBTX_FLASH.ld | *main.c | stm32l4xx_hal_gpio.c

```
388   */
389
390 /**
391   * @brief  Read the specified input port pin.
392   * @param  GPIOx where x can be (A..H) to select the GPIO peripheral for STM32L4 family
393   * @param  GPIO_Pin specifies the port bit to read.
394   *         This parameter can be any combination of GPIO_Pin_x where x can be (0..15).
395   * @retval The input port pin value.
396   */
397 GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
398 {
399   GPIO_PinState bitstatus;
400
401   /* Check the parameters */
402   assert_param(IS_GPIO_PIN(GPIO_Pin));
403
404   if ((GPIOx->IDR & GPIO_Pin) != 0x00u)
405   {
406     bitstatus = GPIO_PIN_SET;
407   }
408   else
409   {
410     bitstatus = GPIO_PIN_RESET;
411   }
412   return bitstatus;
413 }
414
415 /**
416   * @brief  Set or clear the selected data port bit.
417   *
418   * @note   This function uses GPIOx_BSRR and GPIOx_BRR registers to allow atomic read/modify
419   *         accesses. In this way, there is no risk of an IRQ occurring between
420   *         the read and the modify access.
421   *
422   * @param  GPIOx where x can be (A..H) to select the GPIO peripheral for STM32L4 family
423   * @param  GPIO_Pin specifies the port bit to be written.
424   *         This parameter can be any combination of GPIO_Pin_x where x can be (0..15).
425   * @param  PinState specifies the value to be written to the selected bit.
426   *         This parameter can be one of the GPIO_PinState enum values:
427   *            @arg GPIO_PIN_RESET: to clear the port pin
428   *            @arg GPIO_PIN_SET: to set the port pin
429   * @retval None
430   */
431 void HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
432 {
433   /* Check the parameters */
434   assert_param(IS_GPIO_PIN(GPIO_Pin));
435   assert_param(IS_GPIO_PIN_ACTION(PinState));
436
437   if(PinState != GPIO_PIN_RESET)
438   {
439     GPIOx->BSRR = (uint32_t)GPIO_Pin;
```

Console | Problems | Executables | Debugger Console | Memory | Search

Monitors: 0x10000000

0x10000000 : 0x10000000 <Hex> | New Renderings...

Address	0 - 3	4 - 7	8 - B	C - F
10000000	C092C27D	4F412861	4397FE4F	DFED412B
10000010	50BC81CB	AA2067CD	6244F7FB	FAC76A94
10000020	57CC8CD6	55E17C6A	2CF8A333	8F25034A
10000030	3CD69367	27EF1EAD	FA9677DE	D6242145
10000040	8492CC8F	5C2F4580	83800BAE	0D88CA98
10000050	400038D8	EE5452E0	64F92753	3ACC9AB3
10000060	0ADCF4EF	0336AAE1	9489E606	BC540A69
10000070	890FCBD3	0825D26D	B410B43B	DF5DC146

Step 6 : Push the user button

Just put the blue button on your board to activate the write protection as we previously said. If you put the code in pause and try to modify the memory like step 6,you should now receive an error message that the memory is blocked.