STM32 Write Protection

This tutorial will demonstrate the memory write protection mechanism (WRP).

- Flash memory write protection is designed to prevent unwanted write access to defined areas in Flash memory
- The protected area is defined on a per-sector basis.

For STM32L4: the WRP area is defined by “start” and “end” addresses.

Readout protection (RDP) – a global flash memory protection allowing the embedded firmware code to be protected against copy, reverse engineering, dumping, using debug tools or code injection in SRAM.

Most STM32 have 3 levels of Readout protection:

- Level 0: no protection, factory default)
- Level 1: Memory protected (boot mode ≠ Flash memory)
- Level 2: Locked Device

Hardware:

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

Literature:

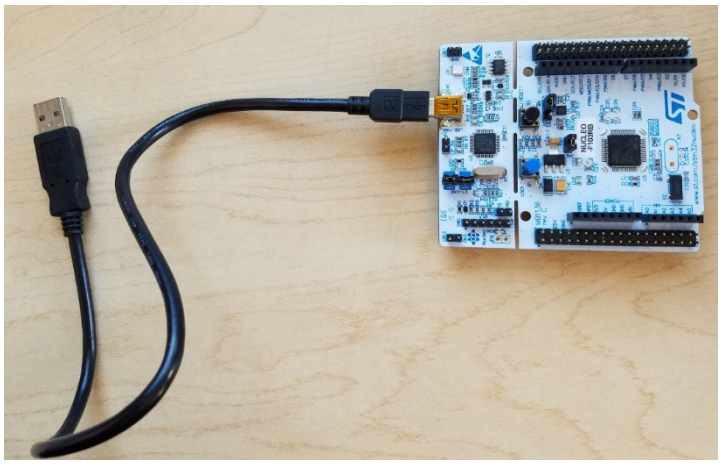
- STM32L476xx Datasheet
- UM1724 User manual STM32 Nucleo-64 boards
- UM1884 Description of STM32L4/L4+ HAL and low-layer drivers
- UM1718 User manual STM32CubeMX for STM32 configuration and initialization C code generation
- RM0351 Reference Manual

Stages

- 1: Connect device
- 2: Launch Cube Programmer
- 3: Add Memory protection
- 4: SRAM2 WRP

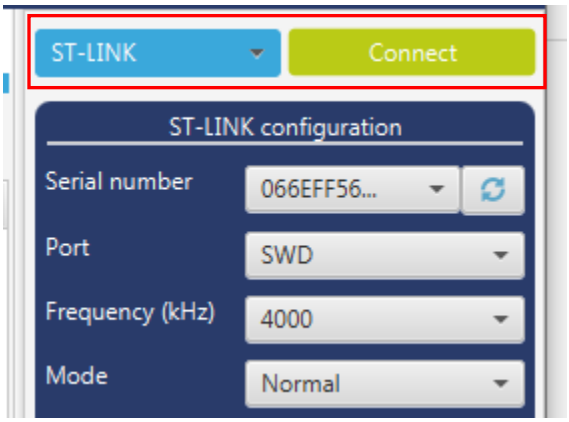
1: Connect device:

Any code may be loaded on to the device. For this tutorial a basic Blinky example is on the board.



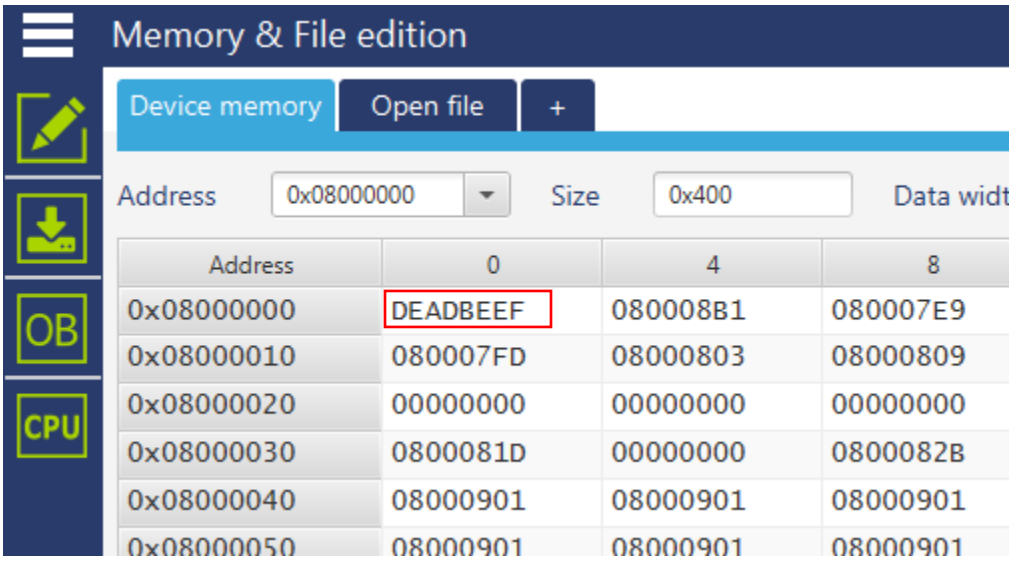
2: Launch Cube Programmer

Launch STM32CubeProgrammer. Select ST-Link and click connect. The flash memory will be shown.



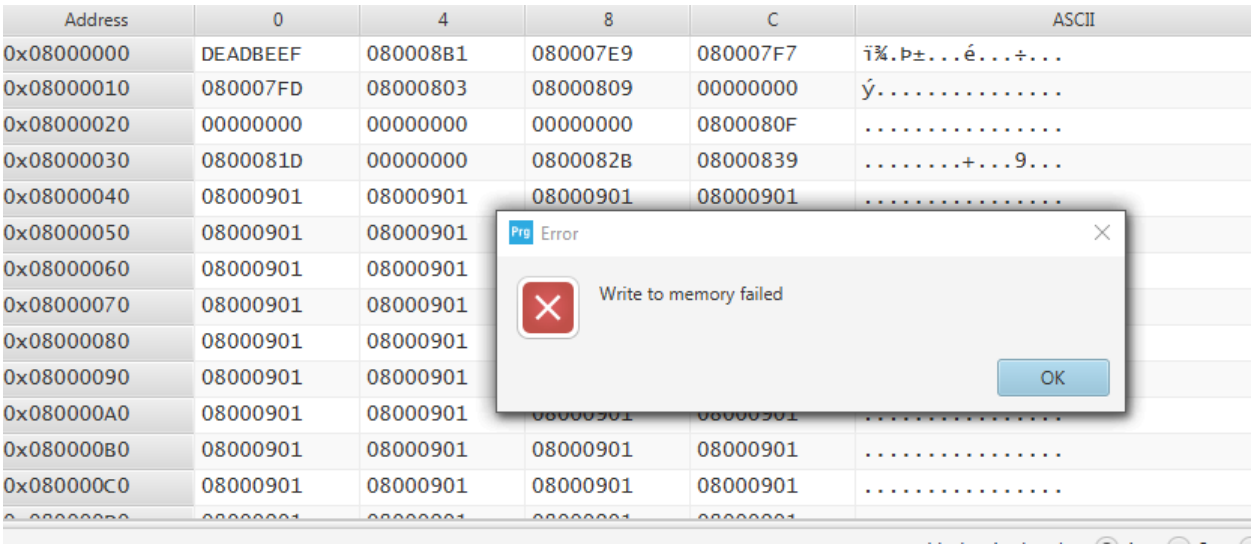
3: Add memory protection

First we will change a value in the flash memory. Modify at address 0x08000000 as show below:



As demonstrated, we can write to the flash memory of the board. Let's add memory protection now. Navigate to Option bytes from the left and select the write protection bank 1 tab. Change WRP1A_STRT to 0x0 and click apply.

Go back to the Memory and File edition and rewrite a value in the same area previously modified. You will see when trying to modify a write to memory failed error will be shown.



To remove the protection change WRP1A_STRT to 0xff and apply. The protection has been removed.

4: SRAM2 WRP

First check the reference manual for SRAM2 address.

SRAM2 Write protection

The SRAM2 can be write protected with a page granularity of 1 Kbyte.

Table 3. SRAM2 organization

Page number	Start address	End address
Page 0	0x1000 0000	0x1000 03FF
Page 1	0x1000 0400	0x1000 07FF
Page 2	0x1000 0800	0x1000 0BFF
Page 3	0x1000 0C00	0x1000 0FFF

SRAM2 address: 0x1000 0000.

Next, launch STM32CubeIDE and generate a project for NucleoL476RG. Go to File->New->STM32 Project. Select NucleoL476RG from board selector. Select “yes” when prompted if peripherals should be initalized in their default mode.

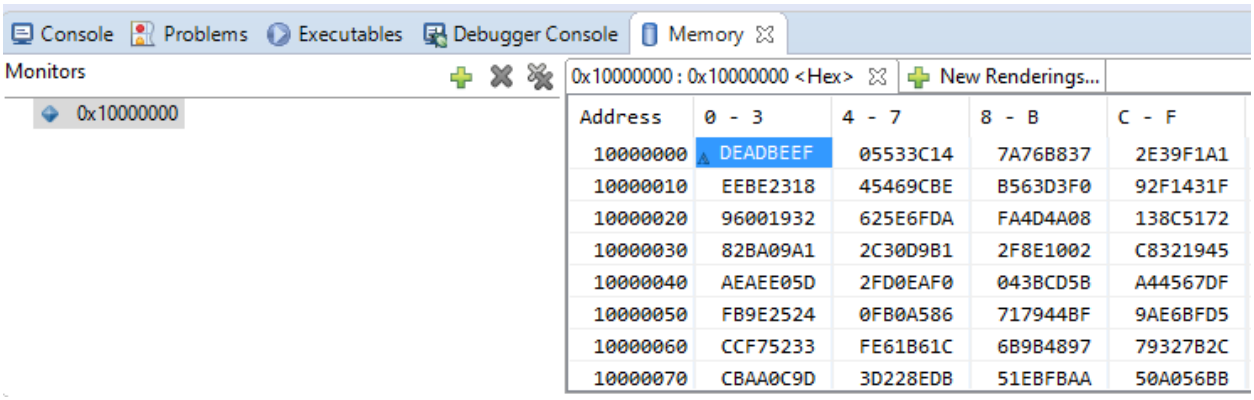
Navigate to main.c. In the main function while(1) loop add:

```
if(HAL_GPIO_ReadPin(B1_GPIO_Port,B1_Pin)  ==  0){
        SYSCFG->SWPR = 0x1;
}
```

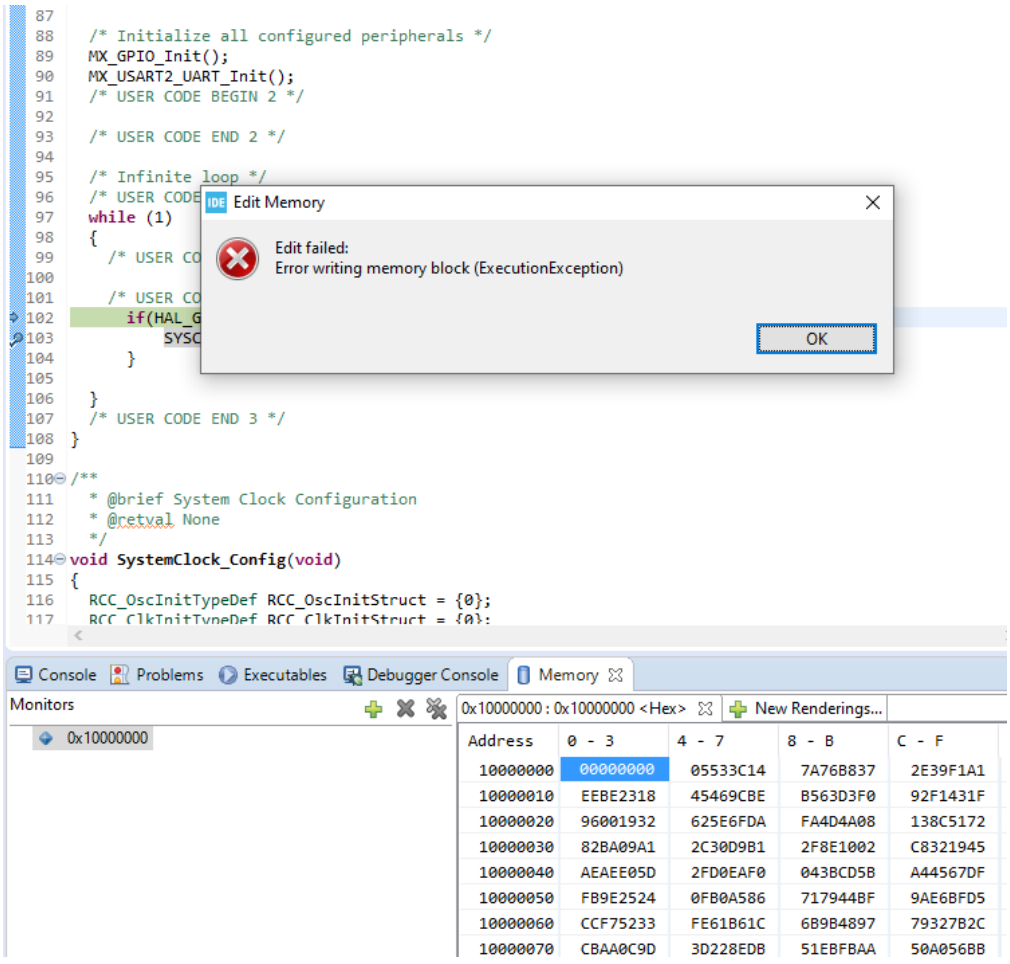
Build the project and start the debug session.Do not resume the execution yet. Add a breakpoint as shown below:

```
 96  /* USER CODE BEGIN WHILE */
 97  while (1)
 98  {
 99    /* USER CODE END WHILE */
100
101    /* USER CODE BEGIN 3 */
102      if(HAL_GPIO_ReadPin(B1_GPIO_Port,B1_Pin)  ==  0){
103          SYSCFG->SWPR = 0x1;
104      }
105
106  }
107  /* USER CODE END 3 */
108 }
```

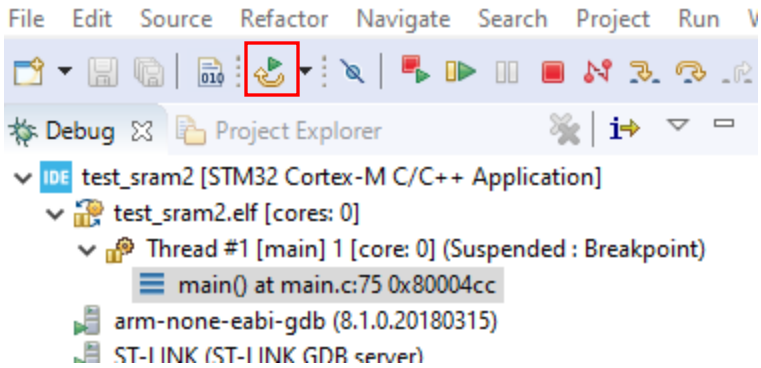
Resume the execution. Let it run for a few seconds and pause the execution. Go to Window->Show View -> Memory. In the memory window, click the + icon and enter the SRAM2 address: 0x1000000. Enter some value as shown below to demonstrate the memory is writable currently.



Continue the execution, but this time press the blue button. It will halt at our breakpoint. At this point we can still modify the memory. Use the step over button to step over the code. Try to write into the memory again. You will be unable to write to the memory.



To reset the board, click the button highlighted in the picture below. This will reset the chip and restart the debug session.



Note: WRP flash protection in combination with RDP level 2 is equivalent to ROM.