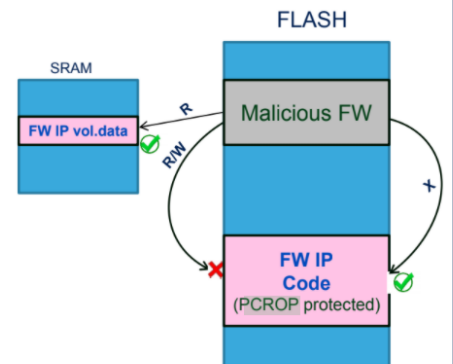


STM32 PCROP

This tutorial will introduce the STM32 PCROP feature. Specifically, we will hide a call to led blinking API in PCROP region.

- The CPU can only jump to PCROP area but cannot read or write it.
- Cannot be read via debugging link

In the photo on the right, the malicious firmware cannot read or write the FW IP. It can however launch it. If launched volatile data will be present in the SRAM and Malicious FW can read it.



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 fw_to_protect.c
- 2: Modify the Compilation option
- 3: Modify the Id file
- 4: Activate PCROP
- 5: Check Protection
- 6: Remove PCROP with STM32CubeProgrammer



1: Create fw_to_protect.c

Launch STM32CubeIDE. Go to *File->New->STM32 Project*. Go to board selector and select Nucleo-L476RG. Select yes when prompted to initialize peripherals in their default mode.

Navigate to the project in the project explorer. Go to *Core->Src*, we will create our *led_blinking.c* file here. Write click the Src folder and select *New->Source File*. Name it *led_blinking.c*

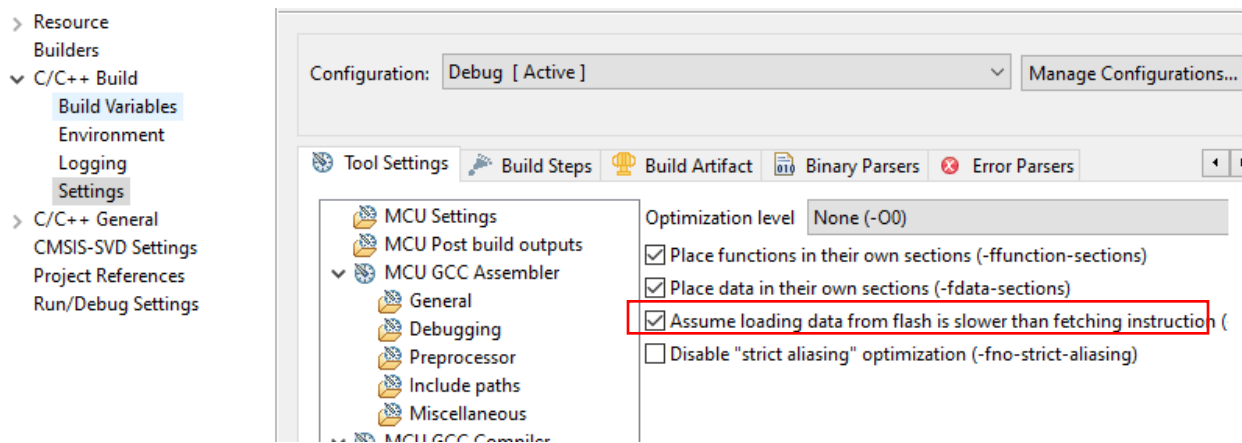
Modify the *led_blinking.c* file as shown below:



```
1  /*
2   * led_blinking.c
3   *
4   * Created on: Nov 15, 2020
5   * Author: faaris
6   */
7
8  #include "main.h"
9
10 uint32_t cpt;
11
12 void secret_led_blinking(void){
13     cpt++;
14     HAL_GPIO_TogglePin(LD2_GPIO_Port,LD2_Pin);
15 }
16
```

2: Modify the compilation option

Next, we want to modify the linker script to put the executable code in *led_blinking.c* at a specific location. Right click the project in Project Explorer and select properties. Go to C/C++ build->Settings. Go to Tool Settings and click on Optimization. Check the box next to "Assume loading data from flash is slower than fetching instruction."



2: Modify the ld file

Now we will modify the linker script. First we have to declare a new region that will be protected. Double click on the STM32L476RGTX_FLASH.ld file. Change the memory definition section to match the image below. Verify that every line is identical to this photo shown here:

```
7 /* Memories definition */
8 MEMORY
9 {
0  RAM      (xrw)      : ORIGIN = 0x20000000,   LENGTH = 96K
1  RAM2     (xrw)      : ORIGIN = 0x10000000,   LENGTH = 32K
2  FLASH    (rx)       : ORIGIN = 0x80000000,   LENGTH = 16K
3  PCROP     (x)        : ORIGIN = 0x80080000, Length = 16K
4 }
5
```

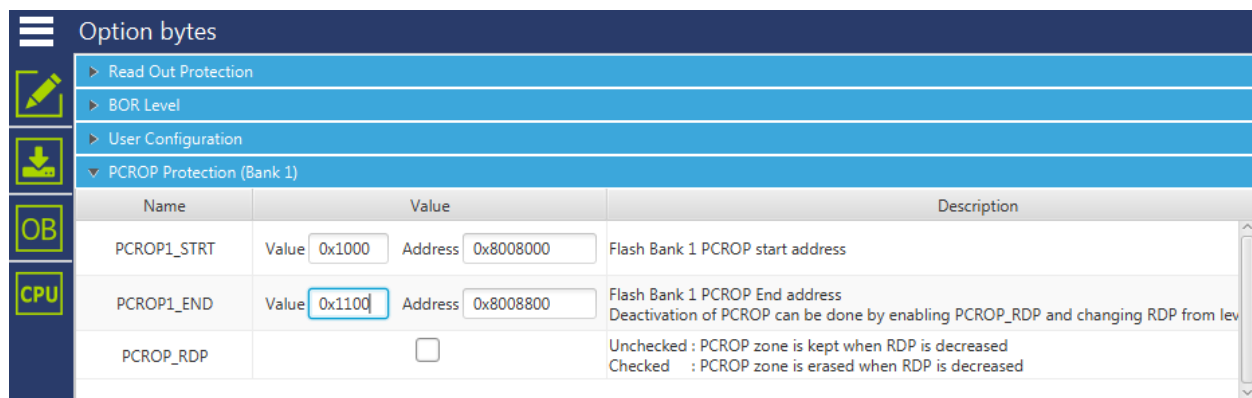
Next, we will define our new section. Add what is shown in the red box:

```
6 /* SECTIONS */
7 SECTIONS
8 {
9  /* The startup code into "FLASH" Rom type memory */
0  .isr_vector :
1  {
2      . = ALIGN(4);
3      KEEP(*(.isr_vector)) /* Startup code */
4      . = ALIGN(4);
5  } >FLASH
6
7  .PCROPed :
8  {
9      . = ALIGN(4);
0      *led_blinking.o (.text .text*)
1      . = ALIGN(4);
2  } > PCROP
3
4  /* The program code and other data into "FLASH" Rom type memory */
5  .text :
6  {
```

Build and start the Debug session of the project. Resume the execution. We can verify that our code blinks the led. Next we will use STM32CubeProgrammer to activate PCROP. Go to Window->Show View->Disassembly. We can see the disassembled code here.

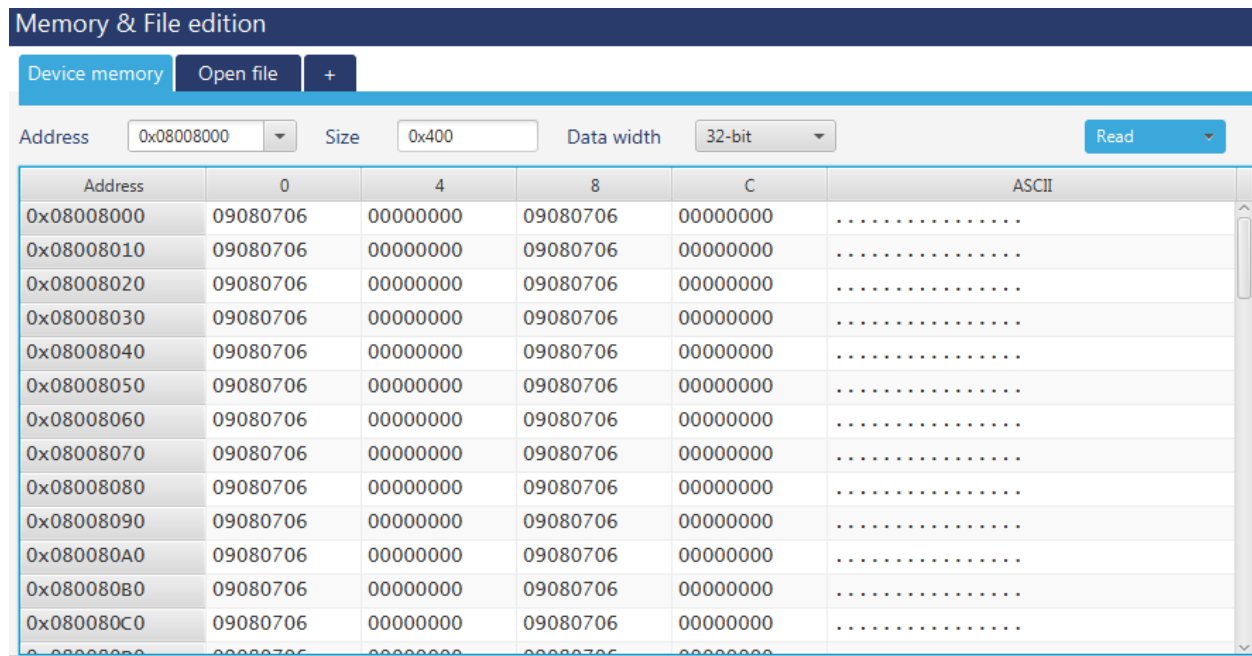
4: Activate PCROP

Open STM32CubeProgrammer. Select ST-LINK and click connect. Navigate to Option Bytes and expand the PCROP Protection (Bank 1). Modify the Value paramter to match the image below:



Click Apply. You will see the LED is no longer blinking. Press the reset button on the board and it will begin blinking again.

Disconnect and reconnect in CubeProgrammer. Navigate to Memory and File tab. We can still see beginning of Flash. Type 0x08008000 into address and press enter. We will see the same value in all locations.

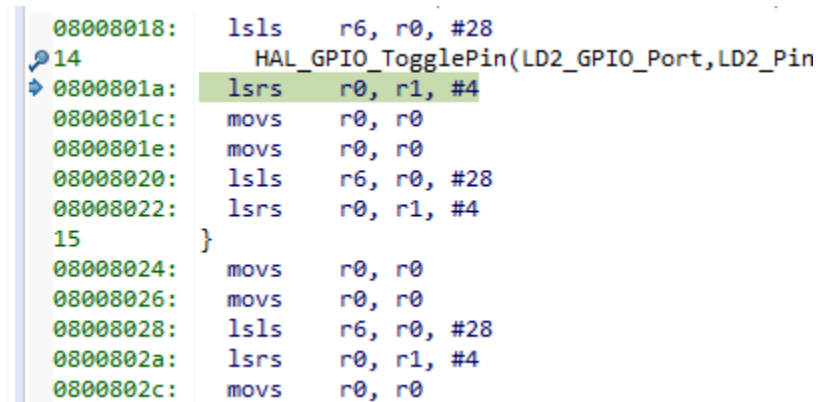


5: Check Protection

Before we could view the disassembled code in STM32CubeIDE. With PCROP activated, we will see if we can repeat that step.

First, when debugging, we want to prevent a redownload of the code since it is already flashed. So right click the project in Project Explorer and go to *Debug As-> Debug Configurations*. Go to the Startup tab and click edit. Uncheck the “Download” option and press okay. Apply and Debug. Remove the breakpoint previously set in *led_blinking.c*. Resume the execution.

We can see the led blinking. Add the breakpoint again and resume. If we view the disassembly window we will see:



```
08008018:  lsls    r6, r0, #28
14      HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin
0800801a:  lsr     r0, r1, #4
0800801c:  movs    r0, r0
0800801e:  movs    r0, r0
08008020:  lsls    r6, r0, #28
08008022:  lsr     r0, r1, #4
15      }
08008024:  movs    r0, r0
08008026:  movs    r0, r0
08008028:  lsls    r6, r0, #28
0800802a:  lsr     r0, r1, #4
0800802c:  movs    r0, r0
```

We can still see instructions; however, these instructions are not the actual executed instructions. We can see our protection is effective. Terminate the debug session.

6: Remove PCROP with STM32CubeProgrammer

Reopen STM32CubeProgrammer and connect again. Go to Option bytes and check the box next to PCROP_RDP and click Apply. Now open the Read Out Protection tab. We need to switch to RDP level 1. Change RDP to BB. And click apply.

Next change RDP to level 0 (AA) and press apply. This action will perform a mass erase of the flash and remove the PCROP region. You will the Option bytes of PCROP Protection (Bank 1) have been reset.

If we look in the flash content we can see everything has been erased.