NEOCRUX
SYSTEMS

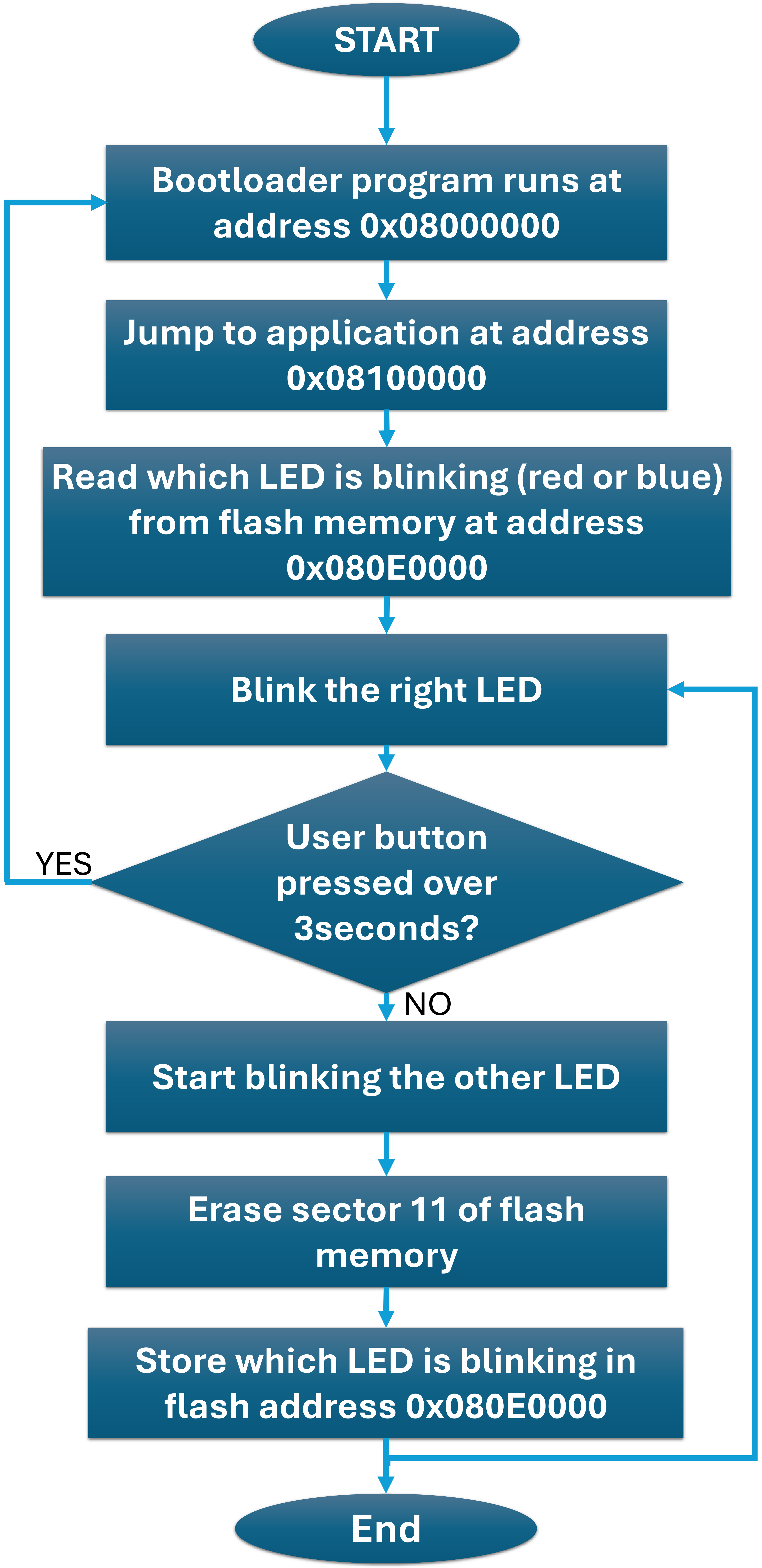
Read, Write and Erase flash memory

Case Study: STM32F767ZI

NEOCRUX SYSTEMS

NEOCRUX SYSTEMS

Diagram of implementation



START

Bootloader program runs at address 0x08000000

Jump to application at address 0x08100000

Read which LED is blinking (red or blue) from flash memory at address 0x080E0000

Blink the right LED

User button pressed over 3seconds?

YES

NO

Start blinking the other LED

Erase sector 11 of flash memory

Store which LED is blinking in flash address 0x080E0000

End

Steps of implementation

STEP1: Understand the FLASH memory architecture from reference manual

- **In this exercice, we will work with dual bank flash**

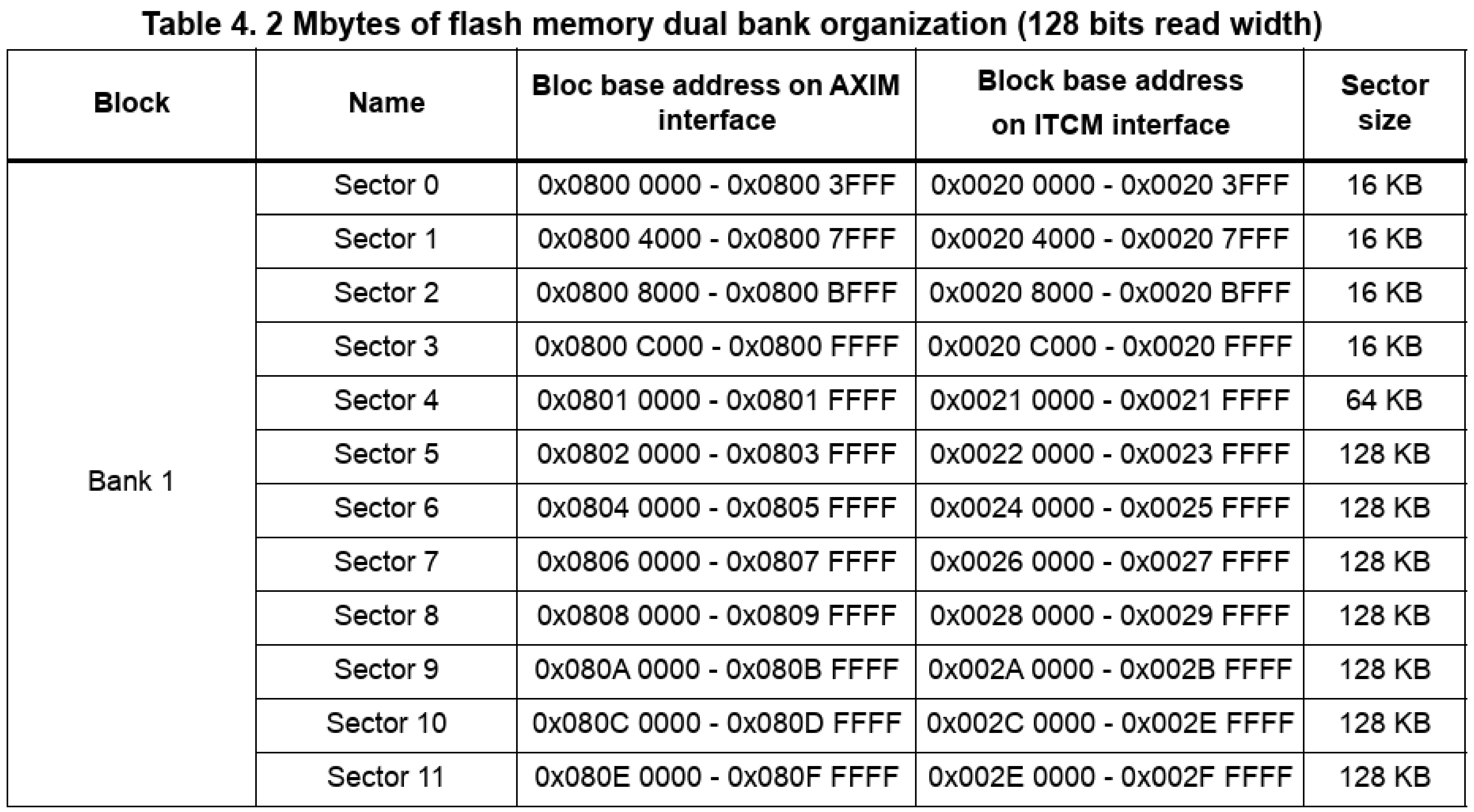


Table 4. 2 Mbytes of flash memory dual bank organization (128 bits read width)

Block	Name	Bloc base address on AXIM interface	Block base address on ITCM interface	Sector size
Bank 1	Sector 0	0x0800 0000 - 0x0800 3FFF	0x0020 0000 - 0x0020 3FFF	16 KB
	Sector 1	0x0800 4000 - 0x0800 7FFF	0x0020 4000 - 0x0020 7FFF	16 KB
	Sector 2	0x0800 8000 - 0x0800 BFFF	0x0020 8000 - 0x0020 BFFF	16 KB
	Sector 3	0x0800 C000 - 0x0800 FFFF	0x0020 C000 - 0x0020 FFFF	16 KB
	Sector 4	0x0801 0000 - 0x0801 FFFF	0x0021 0000 - 0x0021 FFFF	64 KB
	Sector 5	0x0802 0000 - 0x0803 FFFF	0x0022 0000 - 0x0023 FFFF	128 KB
	Sector 6	0x0804 0000 - 0x0805 FFFF	0x0024 0000 - 0x0025 FFFF	128 KB
	Sector 7	0x0806 0000 - 0x0807 FFFF	0x0026 0000 - 0x0027 FFFF	128 KB
	Sector 8	0x0808 0000 - 0x0809 FFFF	0x0028 0000 - 0x0029 FFFF	128 KB
	Sector 9	0x080A 0000 - 0x080B FFFF	0x002A 0000 - 0x002B FFFF	128 KB
	Sector 10	0x080C 0000 - 0x080D FFFF	0x002C 0000 - 0x002E FFFF	128 KB
	Sector 11	0x080E 0000 - 0x080F FFFF	0x002E 0000 - 0x002F FFFF	128 KB

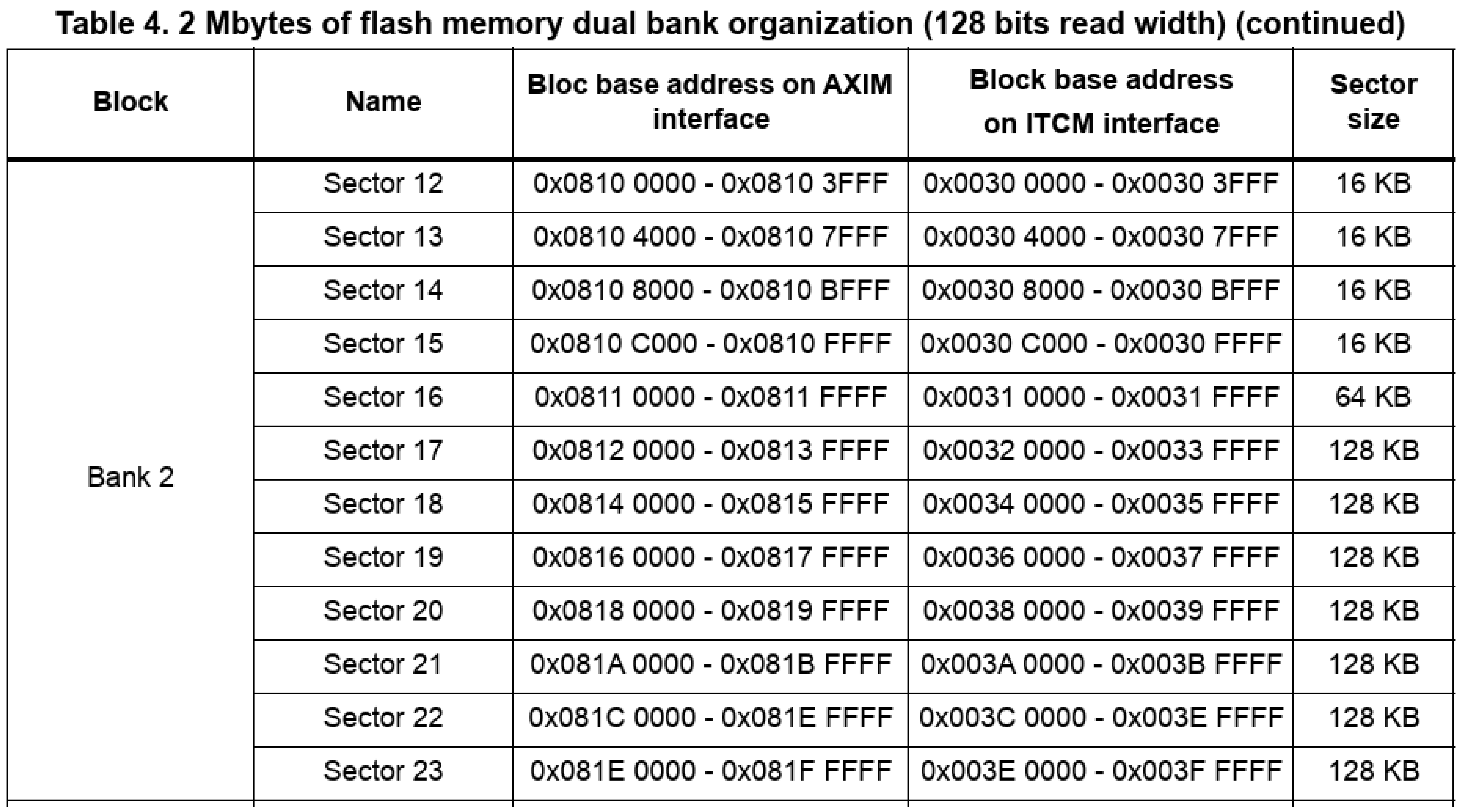


Table 4. 2 Mbytes of flash memory dual bank organization (128 bits read width) (continued)

Block	Name	Bloc base address on AXIM interface	Block base address on ITCM interface	Sector size
Bank 2	Sector 12	0x0810 0000 - 0x0810 3FFF	0x0030 0000 - 0x0030 3FFF	16 KB
	Sector 13	0x0810 4000 - 0x0810 7FFF	0x0030 4000 - 0x0030 7FFF	16 KB
	Sector 14	0x0810 8000 - 0x0810 BFFF	0x0030 8000 - 0x0030 BFFF	16 KB
	Sector 15	0x0810 C000 - 0x0810 FFFF	0x0030 C000 - 0x0030 FFFF	16 KB
	Sector 16	0x0811 0000 - 0x0811 FFFF	0x0031 0000 - 0x0031 FFFF	64 KB
	Sector 17	0x0812 0000 - 0x0813 FFFF	0x0032 0000 - 0x0033 FFFF	128 KB
	Sector 18	0x0814 0000 - 0x0815 FFFF	0x0034 0000 - 0x0035 FFFF	128 KB
	Sector 19	0x0816 0000 - 0x0817 FFFF	0x0036 0000 - 0x0037 FFFF	128 KB
	Sector 20	0x0818 0000 - 0x0819 FFFF	0x0038 0000 - 0x0039 FFFF	128 KB
	Sector 21	0x081A 0000 - 0x081B FFFF	0x003A 0000 - 0x003B FFFF	128 KB
	Sector 22	0x081C 0000 - 0x081E FFFF	0x003C 0000 - 0x003E FFFF	128 KB
	Sector 23	0x081E 0000 - 0x081F FFFF	0x003E 0000 - 0x003F FFFF	128 KB

Steps of implementation

STEP2: Coding game in stm32cube IDE

STEP2.1: As shown in the previous presentation we should:

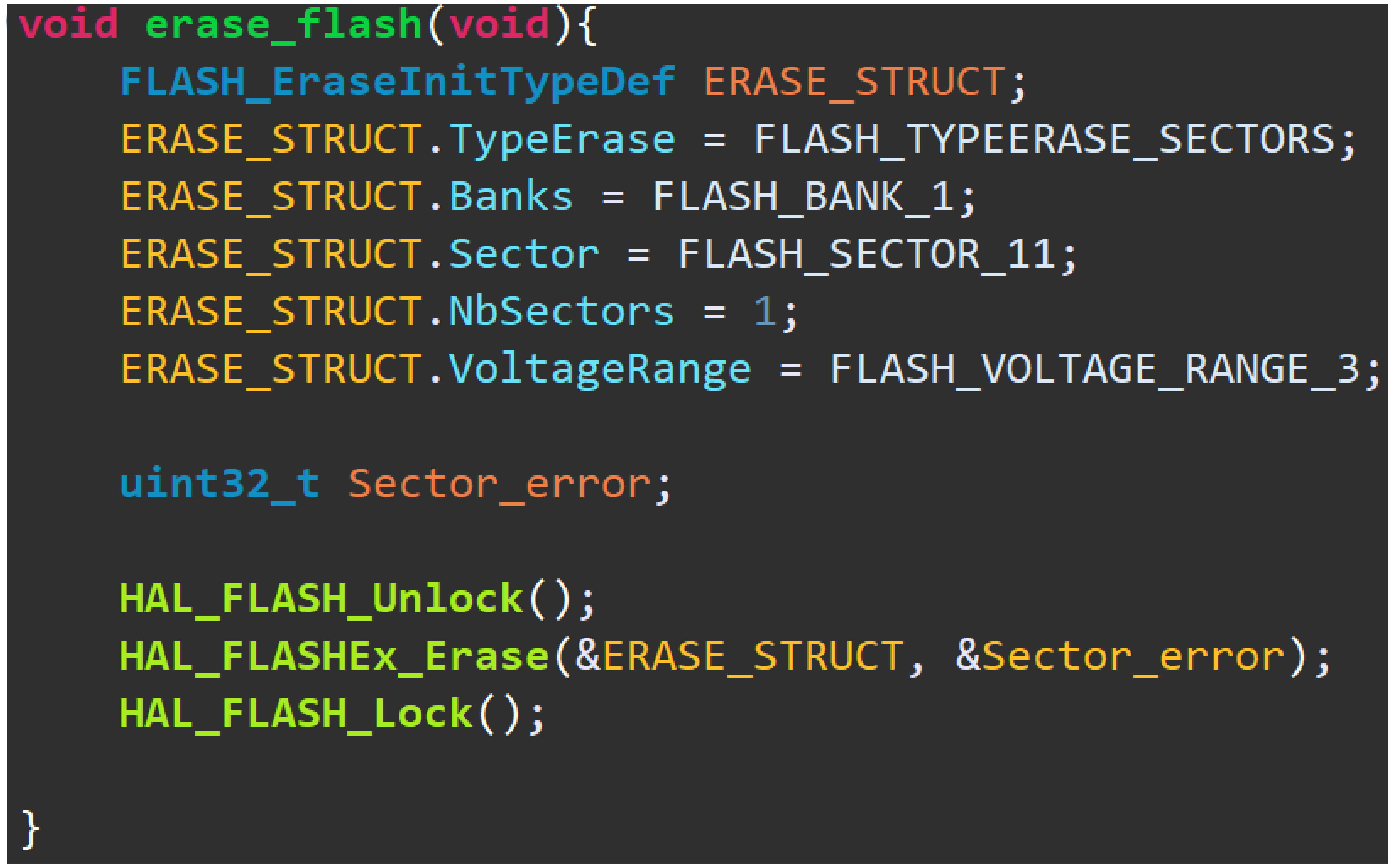
- Create two stm32 projects for bootloader and for application
- Modify the linker files
- Modify vector table base for application
- Create a handler function that allows to jump to bootloader and other to application

- The flash architecture of our project is

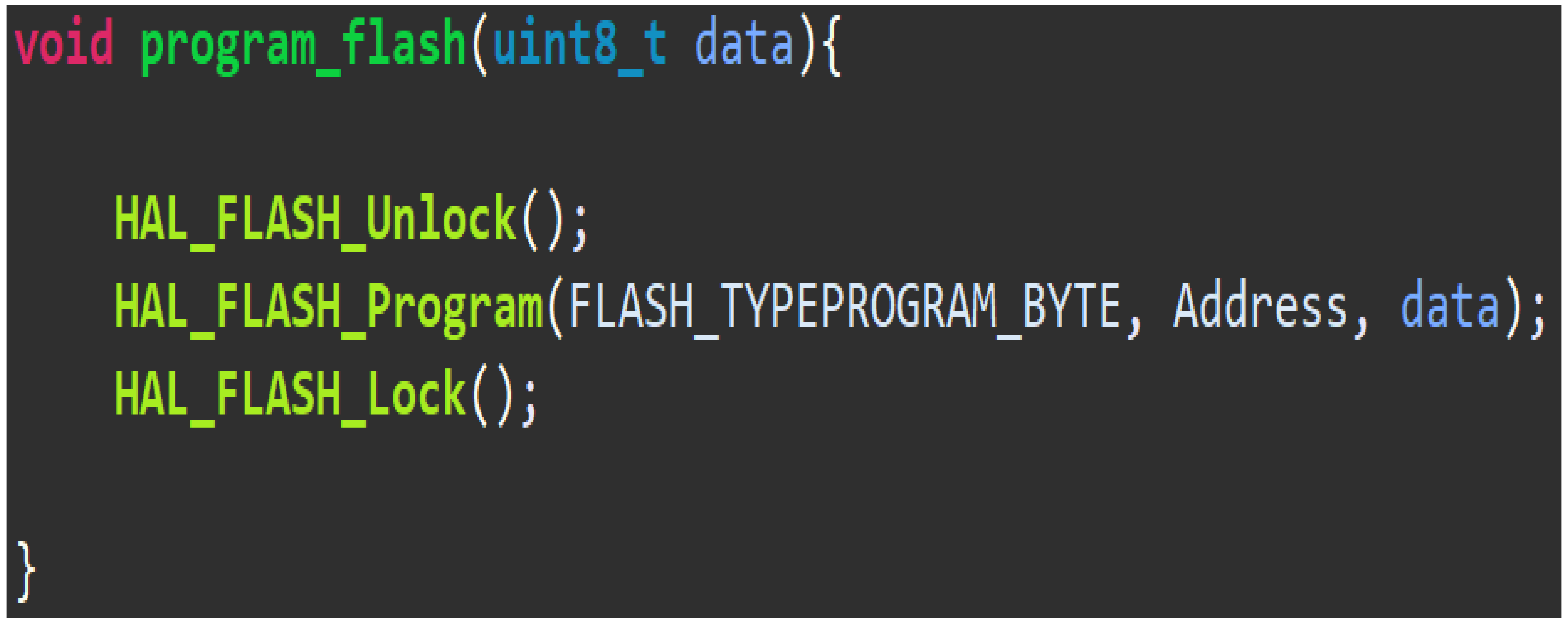
Sectors from 0 to 10	Bootloader
Sector 11	Storing which LED is blinking
Sectors from 12 to 23	Application

Steps of implementation

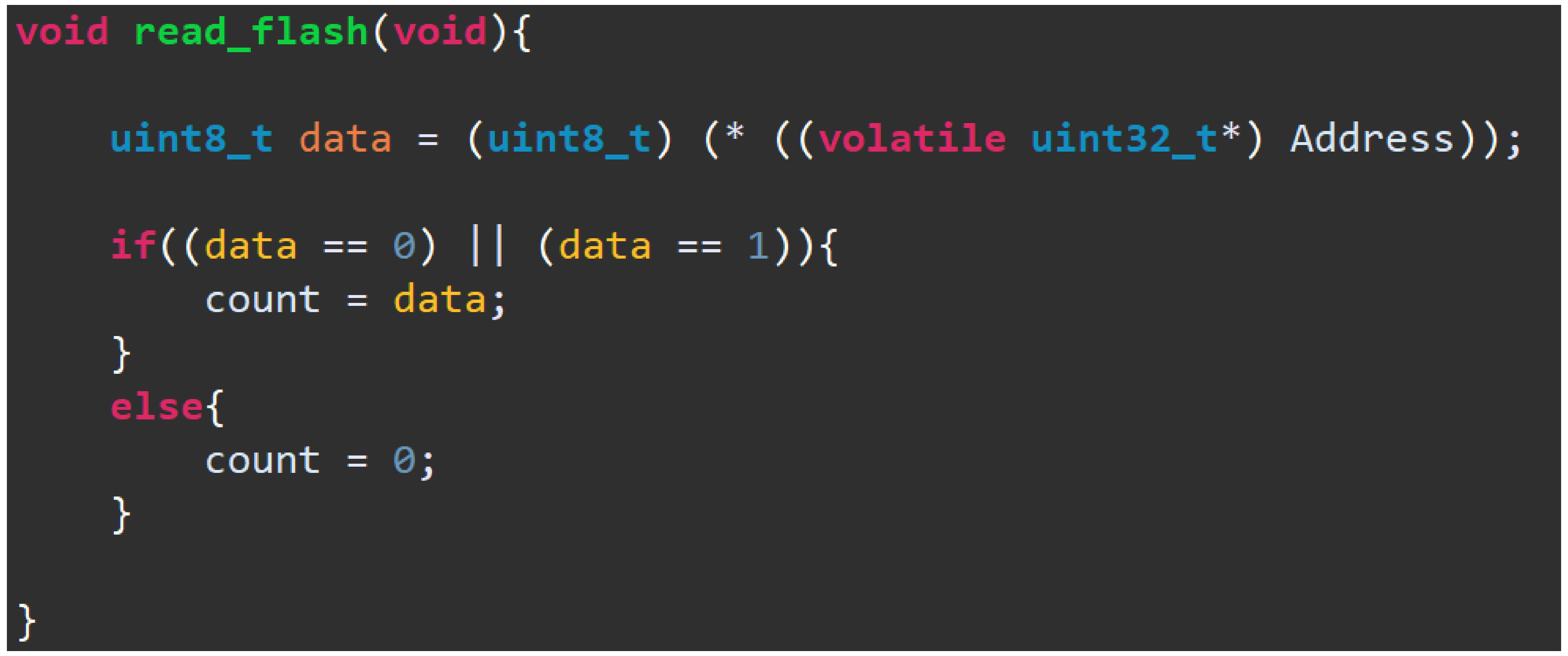
STEP2.2: **Create functions to write, read and erase flash memory**



```
void erase_flash(void){
    FLASH_EraseInitTypeDef ERASE_STRUCT;
    ERASE_STRUCT.TypeErase = FLASH_TYPEERASE_SECTORS;
    ERASE_STRUCT.Banks = FLASH_BANK_1;
    ERASE_STRUCT.Sector = FLASH_SECTOR_11;
    ERASE_STRUCT.NbSectors = 1;
    ERASE_STRUCT.VoltageRange = FLASH_VOLTAGE_RANGE_3;

    uint32_t Sector_error;

    HAL_FLASH_Unlock();
    HAL_FLASHEx_Erase(&ERASE_STRUCT, &Sector_error);
    HAL_FLASH_Lock();

}
```



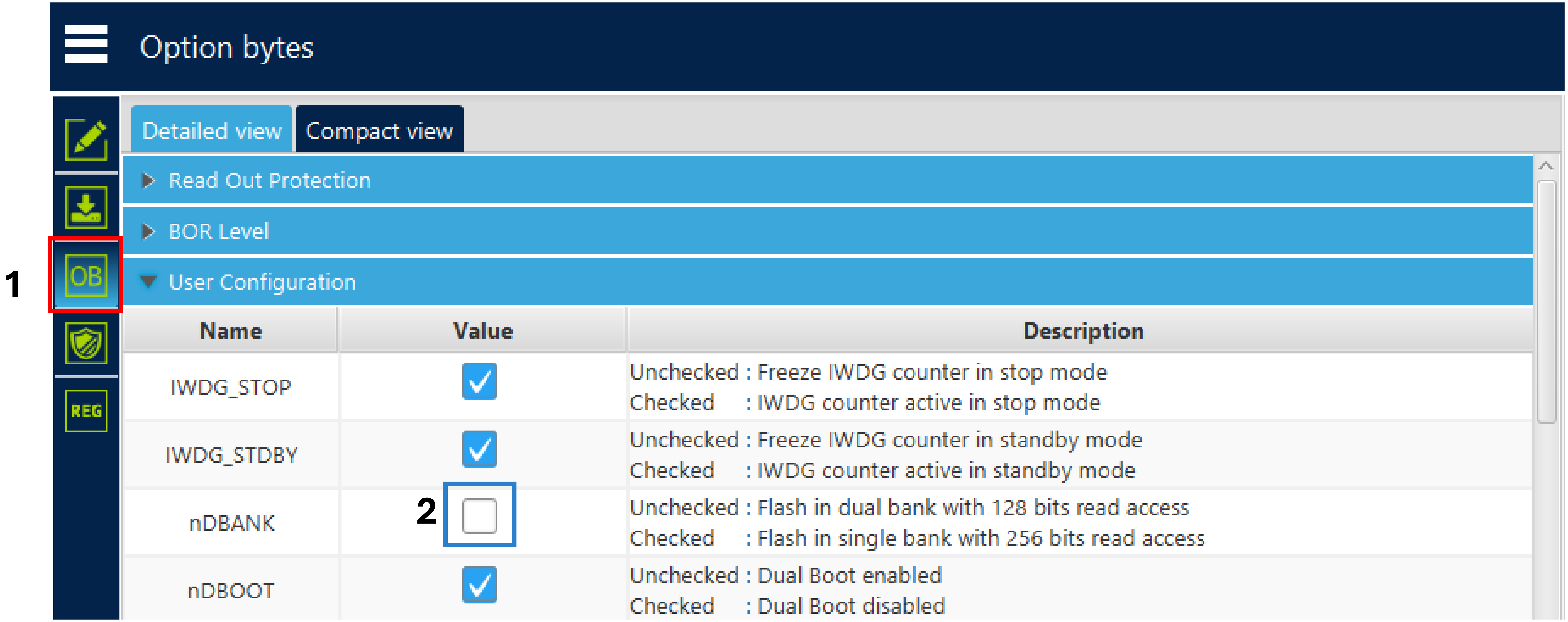
```
void program_flash(uint8_t data){

    HAL_FLASH_Unlock();
    HAL_FLASH_Program(FLASH_TYPEPROGRAM_BYTE, Address, data);
    HAL_FLASH_Lock();

}
```



```
void read_flash(void){

    uint8_t data = (uint8_t) (* ((volatile uint32_t*) Address));

    if((data == 0) || (data == 1)){
        count = data;
    }
    else{
        count = 0;
    }

}
```

Steps of implementation

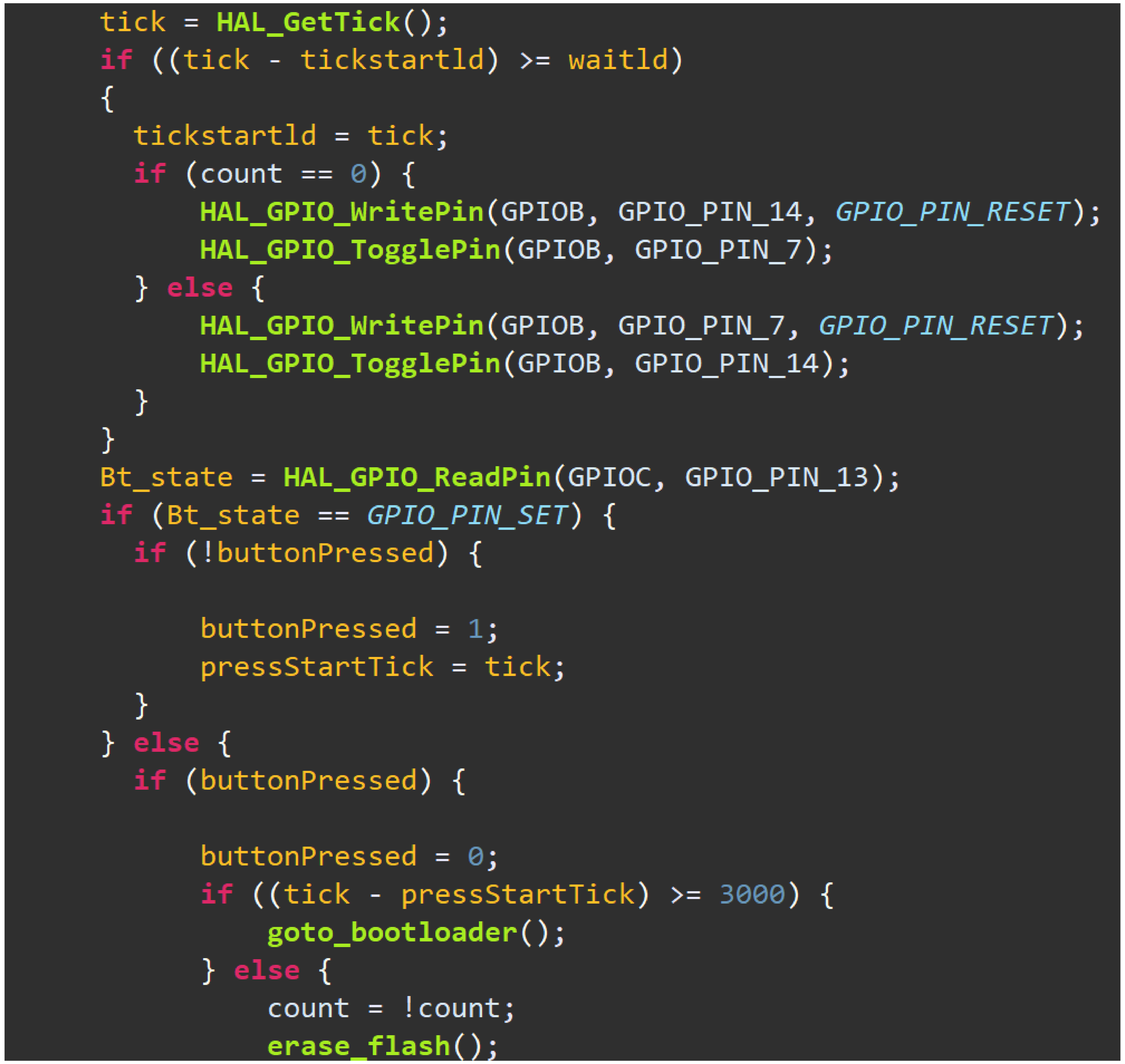
STEP2.3: **Activate dual bank flash from STM32cube programmer**



Name	Value	Description
IWDG_STOP	☑	Unchecked : Freeze IWDG counter in stop mode Checked : IWDG counter active in stop mode
IWDG_STDBY	☑	Unchecked : Freeze IWDG counter in standby mode Checked : IWDG counter active in standby mode
nDBANK	□	Unchecked : Flash in dual bank with 128 bits read access Checked : Flash in single bank with 256 bits read access
nDBOOT	☑	Unchecked : Dual Boot enabled Checked : Dual Boot disabled

Steps of implementation

STEP2.4: **Create application code (A snippet – to see the full code, visit our github)**

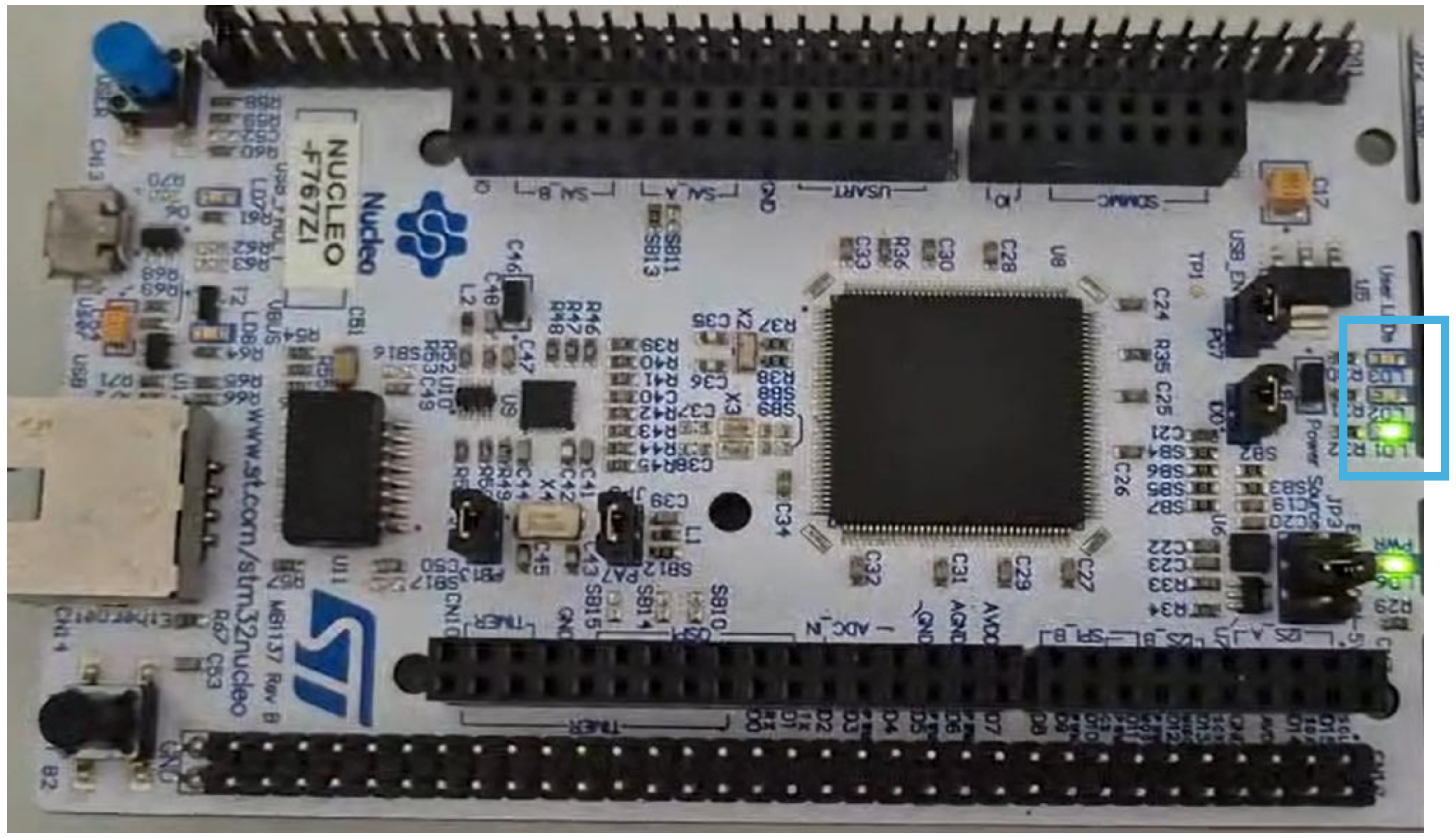


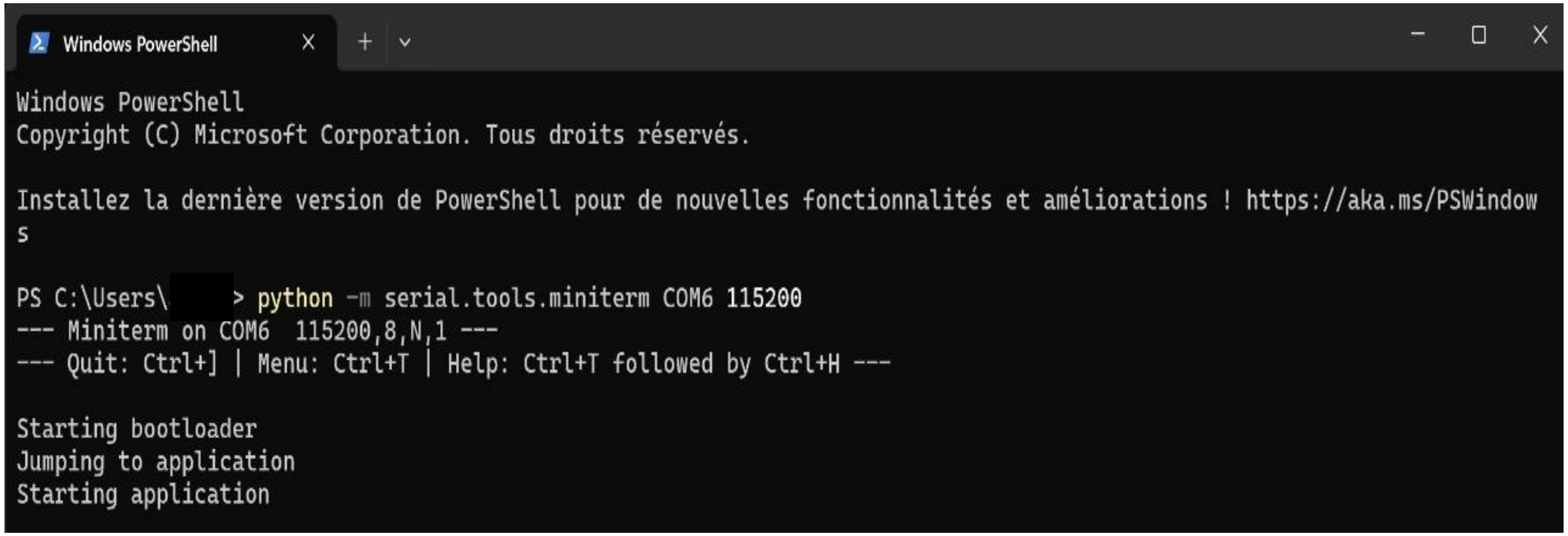
```
tick = HAL_GetTick();
if ((tick - tickstartld) >= waitld)
{
  tickstartld = tick;
  if (count == 0) {
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, GPIO_PIN_RESET);
      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_7);
  } else {
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, GPIO_PIN_RESET);
      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
  }
}
Bt_state = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
if (Bt_state == GPIO_PIN_SET) {
  if (!buttonPressed) {

      buttonPressed = 1;
      pressStartTick = tick;
  }
} else {
  if (buttonPressed) {

      buttonPressed = 0;
      if ((tick - pressStartTick) >= 3000) {
          goto_bootloader();
      } else {
          count = !count;
          erase_flash();
```

Results

After powering the MCU, the CPU goes to the first address of the Flash memory which is 0X08000000.

- **It’s the bootloader**





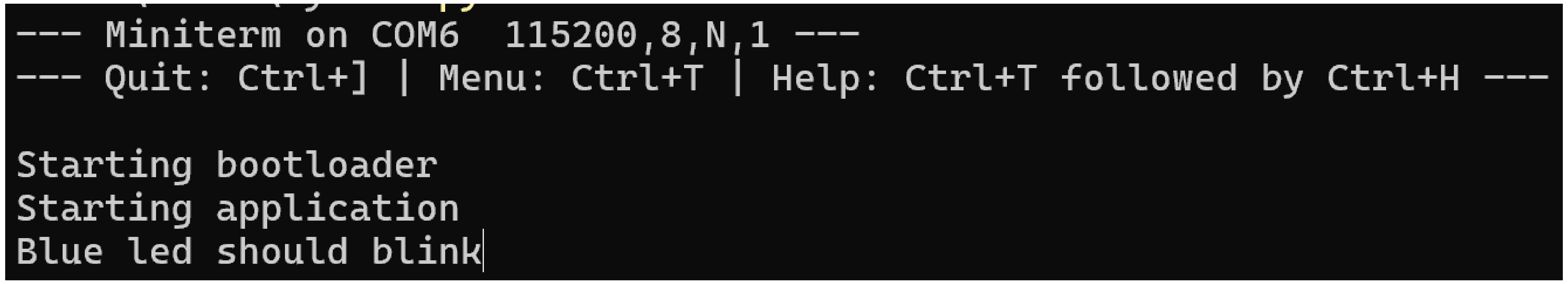
```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Installez la dernière version de PowerShell pour de nouvelles fonctionnalités et améliorations ! https://aka.ms/PSWindow
s

PS C:\Users\[redacted]> python -m serial.tools.miniterm COM6 115200
--- Miniterm on COM6  115200,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

Starting bootloader
Jumping to application
Starting application
```

Results

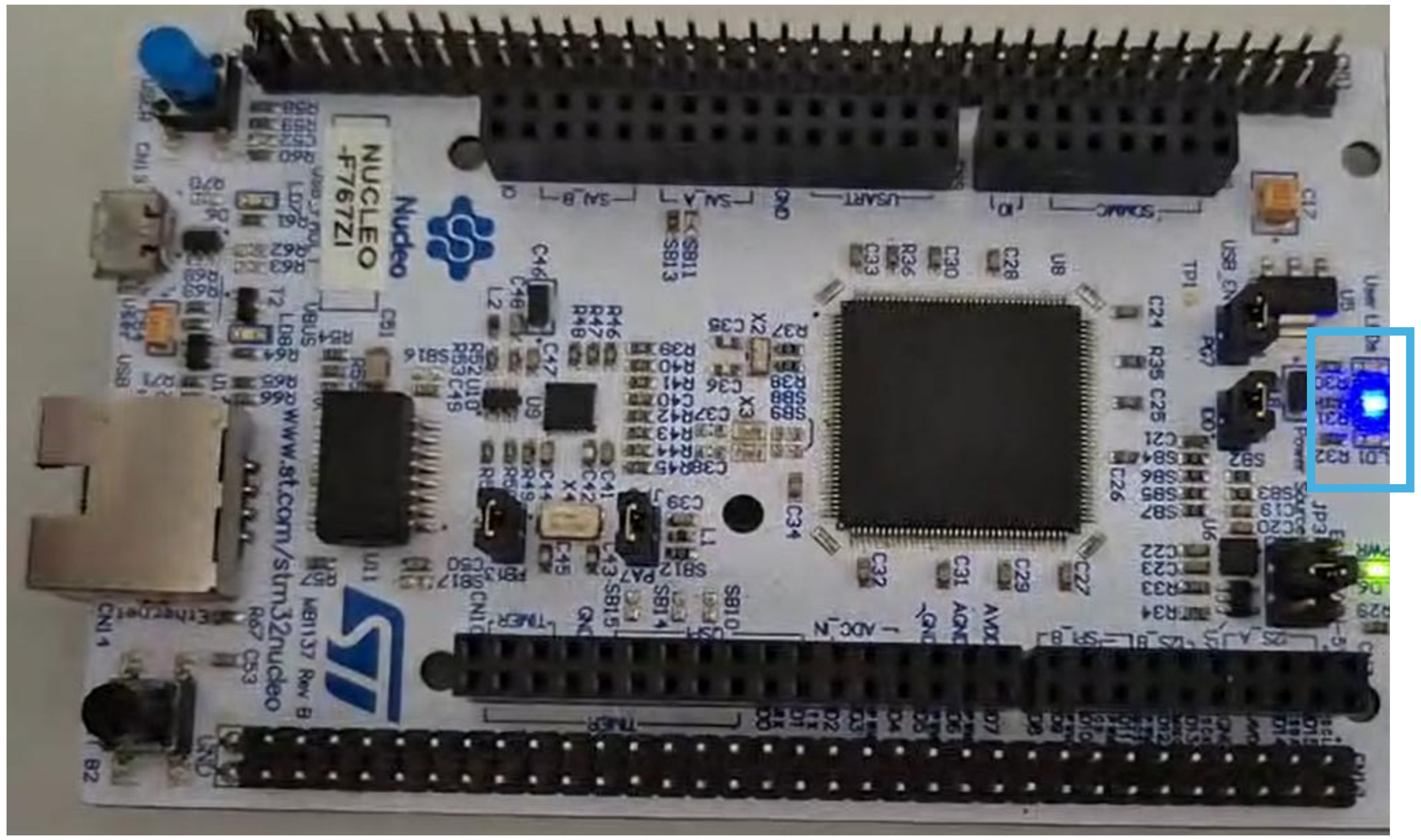
- ➢ Then the bootloader takes care of jumping to the address 0x08100000

  - ▪**It’s the application**

- ➢ The firt thing to do, is reading the flash memory at address 0x080E0000 and read the led that should blink



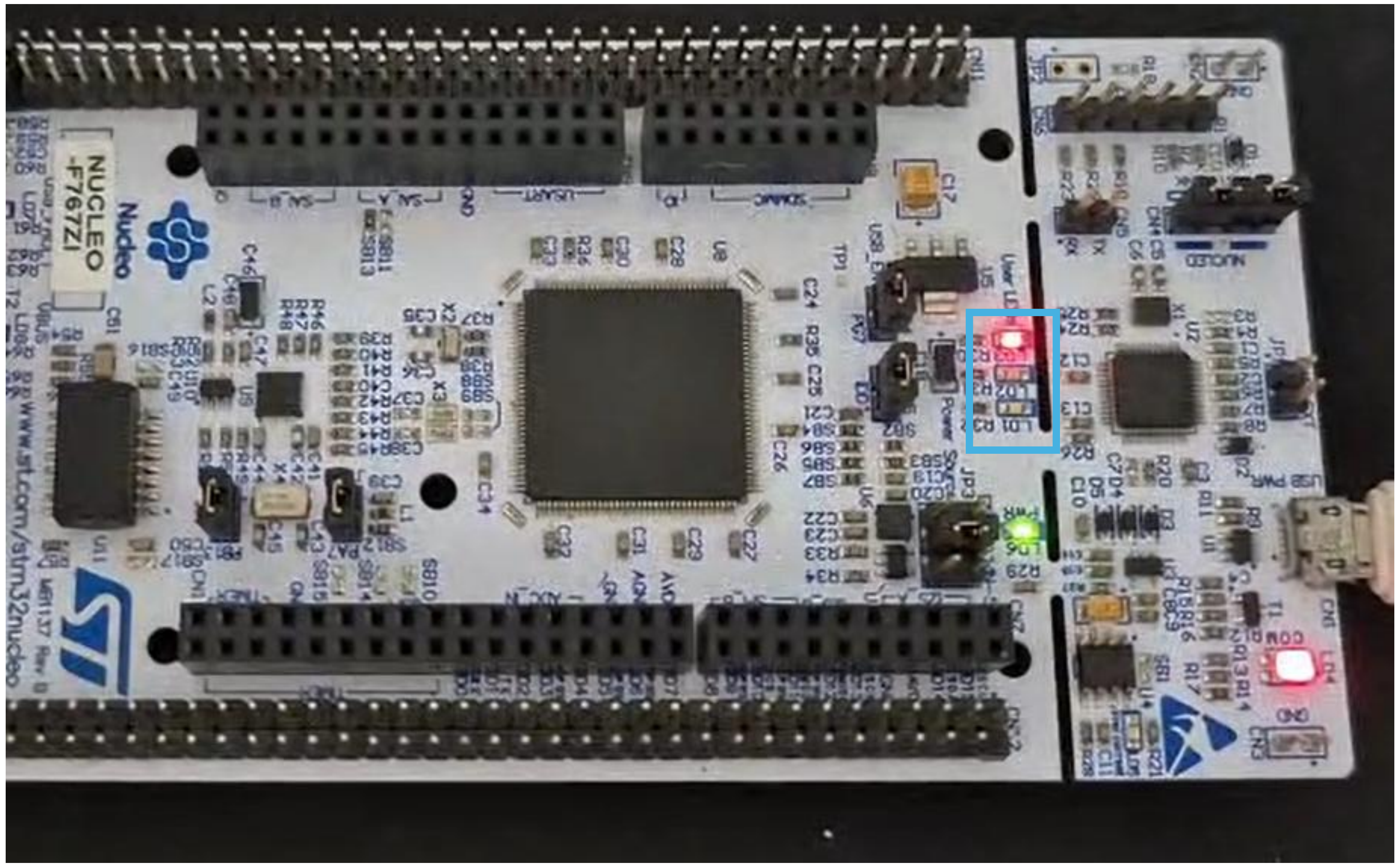
```
--- Miniterm on COM6  115200,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

Starting bootloader
Starting application
Blue led should blink
```



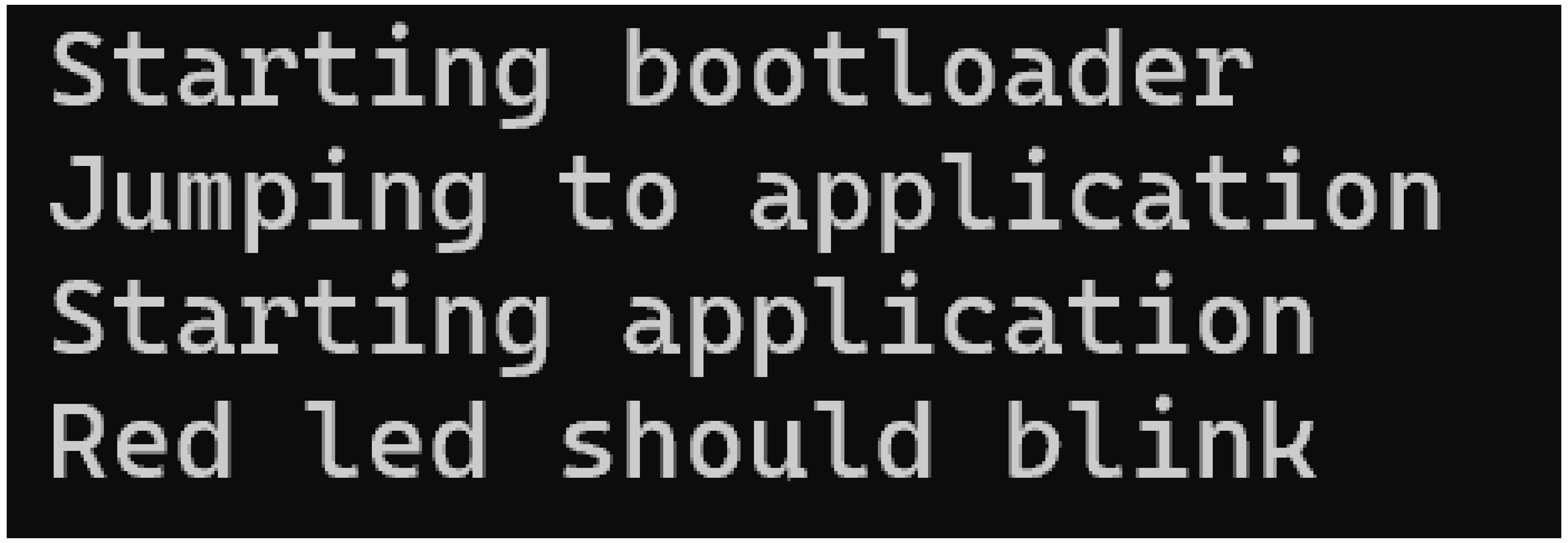
- ➢ And the blue led start blinking

Results

- If we press the user button for less than 3 seconds, the red led should start blinking, and we should store that in flash memory



- Now if we press the user button for more than 3 seconds, the mcu will reset, and the red led should start blinking this time



```
Starting bootloader
Jumping to application
Starting application
Red led should blink
```

See you soon

NEOCRUX
SYSTEMS

NEOCRUX_SYSTEMS
NEOCRUX_SYSTEMS