08/05/2023

Lab 4 - squadra A15

Antonioli Gianfranco 268382
Leli Lorenzo 281728
Ribaudo Alessandro 283309

1.1 - Use switch to switch on/off the LED Low Level Approach

```
/*
 *
 * MODER => set the pin mode (input, output, etc...)
 * ODR => set the value for the pin
 *
 */

int main(void)
{
    //PORT REGISTERS
    // MODER => MODE REGISTER
    // ODR => OUTPUT DATA REGISTER
    // IDR => INPUT DATA REGISTER

    // led pin setup
    volatile unsigned int *GPIOA_MODER = (unsigned int*) (0x40020000 + 0x00);   //
select the correct location of the register (from datasheet)
    volatile unsigned int *GPIOA_ODR = (unsigned int*) (0x40020000 + 0x14);     //
same as above, the offset is 0x14 to get the value (from datasheet)

    // button pin setup
    volatile unsigned int *GPIOC_MODER = (unsigned int*) (0x40020800 + 0x00);
    volatile unsigned int *GPIOC_IDR = (unsigned int*) (0x40020800 + 0x10);

    //CLOCK REGISTERS
    volatile unsigned int *RCC_AHB1ENR = (unsigned int*) (0x40023800 + 0x30);

    //ENABLE PORT CLOCK:
    // this ensure that the peripheral is enabled and connected to the AHB1 bus
    *RCC_AHB1ENR |= 0x05U; // 0x05 = 0b101 to enable port C and A


    //CONFIGURE PORT: set MODER[11:10] = 0x1
    *GPIOA_MODER = *GPIOA_MODER | 0x400;
    *GPIOC_MODER = *GPIOC_MODER | 0x00; // redundant, as the button's default
state is input
    //SWITCH ON THE LED: set ODR[5] = 0x1, that is pulls PA5 high
    //*GPIOA_ODR = *GPIOA_ODR | 0x20;
    // Application code (Infinite loop)
    while (1)
    {
        if(*GPIOC_IDR & 0x2000){
            // turn on
            *GPIOA_ODR = *GPIOA_ODR & ~0x20;
        } else {
            // turn off
            *GPIOA_ODR = *GPIOA_ODR | 0x20;
        }
        //*GPIOA_ODR = *GPIOA_ODR ^ 0x20; // XOR with 1, means "toggle" the bit
    }
}
```

1.2 - Use switch to switch on/off the LED

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */

  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_SYSCFG);
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_PWR);

  NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);

  /* System interrupt 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 */
    int isNotPressed = LL_GPIO_ReadReg(GPIOC, IDR) & 0x2000; // select the 13th
bit (the pushbutton
      if(!isNotPressed){
          LL_GPIO_WriteReg(GPIOA, ODR, LL_GPIO_ReadReg(GPIOA, ODR) | 0x20); //
forces 1 on the 5th bit
      } else {
          LL_GPIO_WriteReg(GPIOA, ODR, LL_GPIO_ReadReg(GPIOA, ODR) & ~0x20); //
forces 0 on the 5th bit
      }
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  LL_FLASH_SetLatency(LL_FLASH_LATENCY_2);
  while(LL_FLASH_GetLatency()!= LL_FLASH_LATENCY_2)
  {
  }
  LL_PWR_SetRegulVoltageScaling(LL_PWR_REGU_VOLTAGE_SCALE2);
  LL_RCC_HSI_SetCalibTrimming(16);
  LL_RCC_HSI_Enable();

   /* Wait till HSI is ready */
  while(LL_RCC_HSI_IsReady() != 1)
  {

  }
  LL_RCC_PLL_ConfigDomain_SYS(LL_RCC_PLLSOURCE_HSI, LL_RCC_PLLM_DIV_16, 336,
LL_RCC_PLLP_DIV_4);
  LL_RCC_PLL_Enable();

   /* Wait till PLL is ready */
  while(LL_RCC_PLL_IsReady() != 1)
  {

  }
  LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
  LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_2);
  LL_RCC_SetAPB2Prescaler(LL_RCC_APB2_DIV_1);
  LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_PLL);

   /* Wait till System clock is ready */
  while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_PLL)
  {

  }
  LL_Init1msTick(84000000);
  LL_SetSystemCoreClock(84000000);
  LL_RCC_SetTIMPrescaler(LL_RCC_TIM_PRESCALER_TWICE);
}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  LL_USART_InitTypeDef USART_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_USART2);

  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  /**USART2 GPIO Configuration
  PA2   ------> USART2_TX
  PA3   ------> USART2_RX
  */
  GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  USART_InitStruct.BaudRate = 115200;
  USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
  USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
  USART_InitStruct.Parity = LL_USART_PARITY_NONE;
  USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
  USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
  USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
  LL_USART_Init(USART2, &USART_InitStruct);
  LL_USART_ConfigAsyncMode(USART2);
  LL_USART_Enable(USART2);
  /* USER CODE BEGIN USART2_Init 2 */

  /* USER CODE END USART2_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  LL_EXTI_InitTypeDef EXTI_InitStruct = {0};
  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOH);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);

  /**/
  LL_GPIO_ResetOutputPin(LD2_GPIO_Port, LD2_Pin);

  /**/
  LL_SYSCFG_SetEXTISource(LL_SYSCFG_EXTI_PORTC, LL_SYSCFG_EXTI_LINE13);

  /**/
  EXTI_InitStruct.Line_0_31 = LL_EXTI_LINE_13;
  EXTI_InitStruct.LineCommand = ENABLE;
  EXTI_InitStruct.Mode = LL_EXTI_MODE_IT;
  EXTI_InitStruct.Trigger = LL_EXTI_TRIGGER_FALLING;
  LL_EXTI_Init(&EXTI_InitStruct);

  /**/
  LL_GPIO_SetPinPull(B1_GPIO_Port, B1_Pin, LL_GPIO_PULL_NO);

  /**/
  LL_GPIO_SetPinMode(B1_GPIO_Port, B1_Pin, LL_GPIO_MODE_INPUT);

  /**/
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
```

```
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

1.3 - Varying blinking frequency

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
#define MYWAIT 32000000 // semiperiodo se lavoriamo a 16Mhz sulle uscite (non sono
sicuro di ciò)
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */

  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_SYSCFG);
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_PWR);

  NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_4);

  /* System interrupt init*/
  /* SysTick_IRQn interrupt configuration */
  NVIC_SetPriority(SysTick_IRQn,
NVIC_EncodePriority(NVIC_GetPriorityGrouping(),15, 0));

  /* 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();
  /* USER CODE BEGIN 2 */
  int count = 1;
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
      if (!LL_GPIO_ReadReg(GPIOC, IDR) | (1 << 13)){
          while (!LL_GPIO_ReadReg(GPIOC, IDR) | (1 << 13)); //aspetta fino a
quando non viene rilasciato il pulsante

          count++;
      }

      // Toggle led
      LL_GPIO_WriteReg(GPIOA, ODR, LL_GPIO_ReadReg(GPIOA, IDR) ^ (1 << 5));
      for (int i = 0; i<MYWAIT/count; i++);
  }
  /* USER CODE END 3 */
```

```
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  LL_FLASH_SetLatency(LL_FLASH_LATENCY_0);
  while(LL_FLASH_GetLatency()!= LL_FLASH_LATENCY_0)
  {
  }
  LL_PWR_SetRegulVoltageScaling(LL_PWR_REGU_VOLTAGE_SCALE2);
  LL_RCC_HSI_SetCalibTrimming(16);
  LL_RCC_HSI_Enable();

   /* Wait till HSI is ready */
  while(LL_RCC_HSI_IsReady() != 1)
  {

  }
  LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
  LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_1);
  LL_RCC_SetAPB2Prescaler(LL_RCC_APB2_DIV_1);
  LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_HSI);

   /* Wait till System clock is ready */
  while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_HSI)
  {

  }
  LL_Init1msTick(16000000);
  LL_SetSystemCoreClock(16000000);
  LL_RCC_SetTIMPrescaler(LL_RCC_TIM_PRESCALER_TWICE);
}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);

  /**/
  LL_GPIO_ResetOutputPin(GPIOA, LL_GPIO_PIN_5);

  /**/
  GPIO_InitStruct.Pin = LL_GPIO_PIN_13;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /**/
  GPIO_InitStruct.Pin = LL_GPIO_PIN_5;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

08/05/2023

2 - Generating a Square Wave

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
```

```
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */

  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_SYSCFG);
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_PWR);

  NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);

  /* System interrupt 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 */
  unsigned int MYWAIT = 3041;
  //SysTick_Config(SystemCoreClock / 1000);
  while (1)
  {
    /* USER CODE END WHILE */
      LL_GPIO_WriteReg(GPIOA, ODR, LL_GPIO_ReadReg(GPIOA, ODR) & ~0x400); //
forces 0 on the 5th bit
      int wave =1;
      for (int i=0; i<MYWAIT; i++){

      }
      LL_GPIO_WriteReg(GPIOA, ODR, LL_GPIO_ReadReg(GPIOA, ODR) | 0x400); // forces
1 on the 5th bit
      wave = 0;
      for (int i=0; i<MYWAIT; i++){

      }
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  LL_FLASH_SetLatency(LL_FLASH_LATENCY_2);
  while(LL_FLASH_GetLatency()!= LL_FLASH_LATENCY_2)
  {
  }
  LL_PWR_SetRegulVoltageScaling(LL_PWR_REGU_VOLTAGE_SCALE2);
  LL_RCC_HSI_SetCalibTrimming(16);
  LL_RCC_HSI_Enable();

   /* Wait till HSI is ready */
  while(LL_RCC_HSI_IsReady() != 1)
  {

  }
  LL_RCC_PLL_ConfigDomain_SYS(LL_RCC_PLLSOURCE_HSI, LL_RCC_PLLM_DIV_16, 336,
LL_RCC_PLLP_DIV_4);
  LL_RCC_PLL_Enable();

   /* Wait till PLL is ready */
  while(LL_RCC_PLL_IsReady() != 1)
  {

  }
  LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
  LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_2);
  LL_RCC_SetAPB2Prescaler(LL_RCC_APB2_DIV_1);
  LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_PLL);

   /* Wait till System clock is ready */
  while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_PLL)
  {
  }
  LL_Init1msTick(84000000);
  LL_SetSystemCoreClock(84000000);
  LL_RCC_SetTIMPrescaler(LL_RCC_TIM_PRESCALER_TWICE);
}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  LL_USART_InitTypeDef USART_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_USART2);

  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  /**USART2 GPIO Configuration
  PA2   ------> USART2_TX
  PA3   ------> USART2_RX
  */
  GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  USART_InitStruct.BaudRate = 115200;
  USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
  USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
  USART_InitStruct.Parity = LL_USART_PARITY_NONE;
  USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
  USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
  USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
  LL_USART_Init(USART2, &USART_InitStruct);
  LL_USART_ConfigAsyncMode(USART2);
  LL_USART_Enable(USART2);
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  LL_EXTI_InitTypeDef EXTI_InitStruct = {0};
  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOH);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);

  /**/
  LL_GPIO_ResetOutputPin(GPIOA, LD2_Pin|LL_GPIO_PIN_10);

  /**/
  LL_SYSCFG_SetEXTISource(LL_SYSCFG_EXTI_PORTC, LL_SYSCFG_EXTI_LINE13);

  /**/
  EXTI_InitStruct.Line_0_31 = LL_EXTI_LINE_13;
  EXTI_InitStruct.LineCommand = ENABLE;
  EXTI_InitStruct.Mode = LL_EXTI_MODE_IT;
  EXTI_InitStruct.Trigger = LL_EXTI_TRIGGER_FALLING;
  LL_EXTI_Init(&EXTI_InitStruct);

  /**/
  LL_GPIO_SetPinPull(B1_GPIO_Port, B1_Pin, LL_GPIO_PULL_NO);

  /**/
  LL_GPIO_SetPinMode(B1_GPIO_Port, B1_Pin, LL_GPIO_MODE_INPUT);

  /**/
  GPIO_InitStruct.Pin = LD2_Pin|LL_GPIO_PIN_10;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
```

```
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```