

Lab 8 - squadra A15

Antonioli Gianfranco 268382

Leli Lorenzo 281728

Ribauda Alessandro 283309

1.1 - Timer Counter Register Polling

```

/* 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);
static void MX_TIM3_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();
    MX_TIM3_Init();

```

```

/* USER CODE BEGIN 2 */
// set timer
LL_TIM_WriteReg(TIM3, ARR, 0xffff); // max value
LL_TIM_WriteReg(TIM3, CR1, LL_TIM_ReadReg(TIM3, CR1) | 0x1); // count enable

/* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
    if (LL_TIM_ReadReg(TIM3, CNT) >= 0x41A0){ // (84 MHz/5 MHz) = 16800 =
0x41A0
        LL_TIM_WriteReg(TIM3, CNT, 0x0000);
        LL_GPIO_WriteReg(GPIOA, ODR, LL_GPIO_ReadReg(GPIOA, ODR) ^ (1 << 10));
// PA10
    }
}
/* 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 TIM3 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM3_Init(void)
{

/* USER CODE BEGIN TIM3_Init 0 */

/* USER CODE END TIM3_Init 0 */

LL_TIM_InitTypeDef TIM_InitStruct = {0};

/* Peripheral clock enable */
LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_TIM3);

/* USER CODE BEGIN TIM3_Init 1 */

/* USER CODE END TIM3_Init 1 */
TIM_InitStruct.Prescaler = 0;
TIM_InitStruct.CounterMode = LL_TIM_COUNTERMODE_UP;
TIM_InitStruct.Autoreload = 65535;
TIM_InitStruct.ClockDivision = LL_TIM_CLOCKDIVISION_DIV1;
LL_TIM_Init(TIM3, &TIM_InitStruct);
LL_TIM_DisableARRPreload(TIM3);
LL_TIM_SetClockSource(TIM3, LL_TIM_CLOCKSOURCE_INTERNAL);
LL_TIM_SetTriggerOutput(TIM3, LL_TIM_TRGO_RESET);
LL_TIM_DisableMasterSlaveMode(TIM3);
/* USER CODE BEGIN TIM3_Init 2 */

/* USER CODE END TIM3_Init 2 */

}

/**
 * @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 */

```

1.2 - Timer Output Compare Flag Polling

```

/* 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 */
#define fclk 84e6    //84 MHz
#define V2p5 (fclk / (2*2.5e3))
#define V12p5 (fclk / (2*12.5e3))
/* 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);
static void MX_TIM3_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();
MX_TIM3_Init();
/* USER CODE BEGIN 2 */
LL_TIM_WriteReg(TIM3, CCR1, V12p5);           // threshold 1
LL_TIM_WriteReg(TIM3, CCR2, V2p5);           // threshold 2
LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~0x2); // delete OC flag
channel 1
LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~0x4); // delete OC flag
channel 2
LL_TIM_WriteReg(TIM3, CR1, LL_TIM_ReadReg(TIM3, CR1) | 0x1); // counter enable
channel 1
LL_TIM_WriteReg(TIM3, CR2, LL_TIM_ReadReg(TIM3, CR2) | 0x1); // counter enable
channel 2

/* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */

    if (LL_TIM_ReadReg(TIM3, SR) & 0x2){
        LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~0x2); // turn off
flag channel 1
        LL_TIM_WriteReg(TIM3, CCR1, LL_TIM_ReadReg(TIM3, CCR1) + V12p5); // set
next threshold
        LL_GPIO_WriteReg(GPIOA, ODR, LL_GPIO_ReadReg(GPIOA, ODR) ^ (1 << 10)); //
toggle pin
    }
    if (LL_TIM_ReadReg(TIM3, SR) & 0x4){
        LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~0x4); // turn off
flag channel 2
        LL_TIM_WriteReg(TIM3, CCR2, LL_TIM_ReadReg(TIM3, CCR2) + V2p5); // set
next threshold
        LL_GPIO_WriteReg(GPIOB, ODR, LL_GPIO_ReadReg(GPIOB, ODR) ^ (1 << 10)); //
toggle pin
    }
}
/* 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 TIM3 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM3_Init(void)
{

    /* USER CODE BEGIN TIM3_Init 0 */

```

```

/* USER CODE END TIM3_Init 0 */

LL_TIM_InitTypeDef TIM_InitStruct = {0};
LL_TIM_OC_InitTypeDef TIM_OC_InitStruct = {0};

/* Peripheral clock enable */
LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_TIM3);

/* USER CODE BEGIN TIM3_Init 1 */

/* USER CODE END TIM3_Init 1 */
TIM_InitStruct.Prescaler = 0;
TIM_InitStruct.CounterMode = LL_TIM_COUNTERMODE_UP;
TIM_InitStruct.Autoreload = 65535;
TIM_InitStruct.ClockDivision = LL_TIM_CLOCKDIVISION_DIV1;
LL_TIM_Init(TIM3, &TIM_InitStruct);
LL_TIM_EnableARRPreload(TIM3);
LL_TIM_SetClockSource(TIM3, LL_TIM_CLOCKSOURCE_INTERNAL);
TIM_OC_InitStruct.OCMode = LL_TIM_OCMODE_ACTIVE;
TIM_OC_InitStruct.OCState = LL_TIM_OCSTATE_ENABLE;
TIM_OC_InitStruct.OCNState = LL_TIM_OCSTATE_DISABLE;
TIM_OC_InitStruct.CompareValue = 5000;
TIM_OC_InitStruct.OCpolarity = LL_TIM_OCPOLARITY_HIGH;
LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH1, &TIM_OC_InitStruct);
LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH1);
TIM_OC_InitStruct.CompareValue = 25000;
LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH2, &TIM_OC_InitStruct);
LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH2);
LL_TIM_SetTriggerOutput(TIM3, LL_TIM_TRGO_RESET);
LL_TIM_DisableMasterSlaveMode(TIM3);
/* USER CODE BEGIN TIM3_Init 2 */

/* USER CODE END TIM3_Init 2 */

}

/**
 * @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_GPIO_ResetOutputPin(GPIOB, 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);

/**/
GPIO_InitStruct.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(GPIOB, &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();
    SysTick_Config(SystemCoreClock / 1000);
    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 - Timer Output Compare Function

```

/* 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 */
#define fclk 84e6    //84 MHz
#define V2p5 (fclk / (2*2.5e3))
#define V12p5 (fclk / (2*12.5e3))
/* 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);
static void MX_TIM3_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();
MX_TIM3_Init();
/* USER CODE BEGIN 2 */
LL_TIM_WriteReg(TIM3, CCR1, V12p5); // threshold 1
LL_TIM_WriteReg(TIM3, CCR2, V2p5); // threshold 2
LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~0x2); // delete OC flag
channel 1
LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~0x4); // delete OC flag
channel 2
LL_TIM_WriteReg(TIM3, CR1, LL_TIM_ReadReg(TIM3, CR1) | 0x1); // counter enable
channel 1
LL_TIM_WriteReg(TIM3, CR2, LL_TIM_ReadReg(TIM3, CR2) | 0x1); // counter enable
channel 2

// SysTick_Config(SystemCoreClock / 1000);
/* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
    if (LL_TIM_ReadReg(TIM3, SR) & 0x2){
        LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~0x2); // turn off
flag channel 1
        LL_TIM_WriteReg(TIM3, CCR1, LL_TIM_ReadReg(TIM3, CCR1) + V12p5); // set
next threshold
    }
    if (LL_TIM_ReadReg(TIM3, SR) & 0x4){
        LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~0x4); // turn off
flag channel 2
        LL_TIM_WriteReg(TIM3, CCR2, LL_TIM_ReadReg(TIM3, CCR2) + V2p5); // set
next threshold
    }
}
/* 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 TIM3 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM3_Init(void)
{

    /* USER CODE BEGIN TIM3_Init 0 */

    /* USER CODE END TIM3_Init 0 */

    LL_TIM_InitTypeDef TIM_InitStruct = {0};
    LL_TIM_OC_InitTypeDef TIM_OC_InitStruct = {0};

```

```

LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

/* Peripheral clock enable */
LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_TIM3);

/* USER CODE BEGIN TIM3_Init 1 */

/* USER CODE END TIM3_Init 1 */
TIM_InitStruct.Prescaler = 0;
TIM_InitStruct.CounterMode = LL_TIM_COUNTERMODE_UP;
TIM_InitStruct.Autoreload = 65535;
TIM_InitStruct.ClockDivision = LL_TIM_CLOCKDIVISION_DIV1;
LL_TIM_Init(TIM3, &TIM_InitStruct);
LL_TIM_DisableARRPreload(TIM3);
LL_TIM_SetClockSource(TIM3, LL_TIM_CLOCKSOURCE_INTERNAL);
TIM_OC_InitStruct.OCMode = LL_TIM_OCMODE_TOGGLE;
TIM_OC_InitStruct.OCState = LL_TIM_OCSTATE_ENABLE;
TIM_OC_InitStruct.OCNState = LL_TIM_OCSTATE_DISABLE;
TIM_OC_InitStruct.CompareValue = 0;
TIM_OC_InitStruct.OCpolarity = LL_TIM_OCPOLARITY_HIGH;
LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH1, &TIM_OC_InitStruct);
LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH1);
LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH2, &TIM_OC_InitStruct);
LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH2);
LL_TIM_SetTriggerOutput(TIM3, LL_TIM_TRGO_RESET);
LL_TIM_DisableMasterSlaveMode(TIM3);
/* USER CODE BEGIN TIM3_Init 2 */

/* USER CODE END TIM3_Init 2 */
LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
/**TIM3 GPIO Configuration
PC6  -----> TIM3_CH1
PC7  -----> TIM3_CH2
*/
GPIO_InitStruct.Pin = LL_GPIO_PIN_6|LL_GPIO_PIN_7;
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_2;
LL_GPIO_Init(GPIOC, &GPIO_InitStruct);

}

/**
 * @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_PUSH_PULL;
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 */

```

1.4 - Timer Output Compare Function with Variable 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 */
#define fmax 4500 // Hz

```

```

#define fmin 800      // Hz
#define fclk 84e6     // Hz
#define potmax 65

/* 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);
static void MX_TIM3_Init(void);
static void MX_ADC1_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();
MX_TIM3_Init();
MX_ADC1_Init();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
LL_ADC_WriteReg(ADC1, CR2, LL_ADC_ReadReg(ADC1, CR2) | ((1 << 30) + (1))); //
set ADON and SWSTART to 1 in order to enable ADC
LL_ADC_WriteReg(ADC1, CR2, LL_ADC_ReadReg(ADC1, CR2) | ((1 << 30) + (1))); //
don't know why, I need to set it TWO times in order to make it work... boh...

LL_TIM_WriteReg(TIM3, CCR1, 0x20D0); // set initial threshold
LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~0x2); // delete OC flag
channel 1
LL_TIM_WriteReg(TIM3, CR1, LL_TIM_ReadReg(TIM3, CR1) | 0x1); // counter enable
channel 1

while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

    if (LL_TIM_ReadReg(TIM3, SR) & 0x2){ // if timer reaches threshold
        LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~0x2); // turn off
flag channel 1
        if(LL_ADC_ReadReg(ADC1, SR) & (1 << 1)){ // read EOC bit: if ADC finishes
conversion
            LL_ADC_WriteReg(ADC1, SR, LL_ADC_ReadReg(ADC1, SR) & ~(1 << 1)); //
reset EOC bit
            int pot = LL_ADC_ReadReg(ADC1, DR) & 0xFFFF; // read pot current value
            float f = fmin + (pot/((float)potmax))*(fmax-fmin);
            int val = fclk / (2*f);
            // for 200K pot:
            // min: 0
            // max: 65
            LL_TIM_WriteReg(TIM3, CCR1, LL_TIM_ReadReg(TIM3, CCR1) + val); // set
next threshold

```

```

    }
}
}
/* 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 ADC1 Initialization Function
 * @param None
 * @retval None
 */

```



```

static void MX_ADC1_Init(void)
{

    /* USER CODE BEGIN ADC1_Init 0 */

    /* USER CODE END ADC1_Init 0 */

    LL_ADC_InitTypeDef ADC_InitStruct = {0};
    LL_ADC_REG_InitTypeDef ADC_REG_InitStruct = {0};
    LL_ADC_CommonInitTypeDef ADC_CommonInitStruct = {0};

    LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

    /* Peripheral clock enable */
    LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_ADC1);

    LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
    /**ADC1 GPIO Configuration
    PA0-WKUP      -> ADC1_IN0
    */
    GPIO_InitStruct.Pin = LL_GPIO_PIN_0;
    GPIO_InitStruct.Mode = LL_GPIO_MODE_ANALOG;
    GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
    LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    /* USER CODE BEGIN ADC1_Init 1 */

    /* USER CODE END ADC1_Init 1 */
    /** Common config
    */
    ADC_InitStruct.Resolution = LL_ADC_RESOLUTION_8B;
    ADC_InitStruct.DataAlignment = LL_ADC_DATA_ALIGN_RIGHT;
    ADC_InitStruct.SequencersScanMode = LL_ADC_SEQ_SCAN_DISABLE;
    LL_ADC_Init(ADC1, &ADC_InitStruct);
    ADC_REG_InitStruct.TriggerSource = LL_ADC_REG_TRIG_SOFTWARE;
    ADC_REG_InitStruct.SequencerLength = LL_ADC_REG_SEQ_SCAN_DISABLE;
    ADC_REG_InitStruct.SequencerDiscont = LL_ADC_REG_SEQ_DISCONT_DISABLE;
    ADC_REG_InitStruct.ContinuousMode = LL_ADC_REG_CONV_CONTINUOUS;
    ADC_REG_InitStruct.DMATransfer = LL_ADC_REG_DMA_TRANSFER_NONE;
    LL_ADC_REG_Init(ADC1, &ADC_REG_InitStruct);
    LL_ADC_REG_SetFlagEndOfConversion(ADC1, LL_ADC_REG_FLAG_EOC_SEQUENCE_CONV);
    ADC_CommonInitStruct.CommonClock = LL_ADC_CLOCK_SYNC_PCLK_DIV4;
    LL_ADC_CommonInit(__LL_ADC_COMMON_INSTANCE(ADC1), &ADC_CommonInitStruct);
    /** Configure Regular Channel
    */
    LL_ADC_REG_SetSequencerRanks(ADC1, LL_ADC_REG_RANK_1, LL_ADC_CHANNEL_0);
    LL_ADC_SetChannelSamplingTime(ADC1, LL_ADC_CHANNEL_0,
    LL_ADC_SAMPLINGTIME_3CYCLES);
    /* USER CODE BEGIN ADC1_Init 2 */

    /* USER CODE END ADC1_Init 2 */

}

```

```

/**
 * @brief TIM3 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM3_Init(void)
{

    /* USER CODE BEGIN TIM3_Init 0 */

    /* USER CODE END TIM3_Init 0 */

    LL_TIM_InitTypeDef TIM_InitStruct = {0};
    LL_TIM_OC_InitTypeDef TIM_OC_InitStruct = {0};

    LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

    /* Peripheral clock enable */
    LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_TIM3);

    /* USER CODE BEGIN TIM3_Init 1 */

    /* USER CODE END TIM3_Init 1 */
    TIM_InitStruct.Prescaler = 0;
    TIM_InitStruct.CounterMode = LL_TIM_COUNTERMODE_UP;
    TIM_InitStruct.Autoreload = 65535;
    TIM_InitStruct.ClockDivision = LL_TIM_CLOCKDIVISION_DIV1;
    LL_TIM_Init(TIM3, &TIM_InitStruct);
    LL_TIM_DisableARRPreload(TIM3);
    LL_TIM_SetClockSource(TIM3, LL_TIM_CLOCKSOURCE_INTERNAL);
    TIM_OC_InitStruct.OCMode = LL_TIM_OCMODE_TOGGLE;
    TIM_OC_InitStruct.OCState = LL_TIM_OCSTATE_ENABLE;
    TIM_OC_InitStruct.OCNState = LL_TIM_OCSTATE_DISABLE;
    TIM_OC_InitStruct.CompareValue = 0;
    TIM_OC_InitStruct.OCpolarity = LL_TIM_OCPOLARITY_HIGH;
    LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH1, &TIM_OC_InitStruct);
    LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH1);
    LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH2, &TIM_OC_InitStruct);
    LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH2);
    LL_TIM_SetTriggerOutput(TIM3, LL_TIM_TRGO_RESET);
    LL_TIM_DisableMasterSlaveMode(TIM3);
    /* USER CODE BEGIN TIM3_Init 2 */

    /* USER CODE END TIM3_Init 2 */
    LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
    /**TIM3 GPIO Configuration
    PC6 ----> TIM3_CH1
    PC7 ----> TIM3_CH2
    */
    GPIO_InitStruct.Pin = LL_GPIO_PIN_6|LL_GPIO_PIN_7;
    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_2;
    LL_GPIO_Init(GPIOC, &GPIO_InitStruct);

}

/**
 * @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 = LL_GPIO_PIN_1;
    GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
    LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    /**/
    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 */

```