```
/* 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.
  *

******************************************************************************
  * Git Repo
  * https://github.com/Naikthelegend/3096_pracs_FRSKIA001_CLRCAM007/
tree/master

******************************************************************************
  */
/* USER CODE END Header */
/* Includes
------------------------------------------------------------------*/
#include "main.h"
#include "adc.h"
#include "tim.h"
#include "gpio.h"

/* Private includes
----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stm32f0xx.h"
#include <lcd_stm32f0.c>
/* 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 */
uint32_t prev_millis = 0;
uint32_t curr_millis = 0;
uint32_t delay_t = 500; // Initialise delay to 500ms
uint32_t adc_val;
uint32_t previous_tick = 0;
/* USER CODE END PV */

/* Private function prototypes
-----------------------------------------------*/
void SystemClock_Config(void);
static void MX_NVIC_Init(void);
/* USER CODE BEGIN PFP */
void EXTI0_1_IRQHandler(void);
void writeLCD(char *char_in);
uint32_t pollADC(void);
uint32_t ADCtoCCR(uint32_t adc_val);
/* 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. */
    HAL_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_ADC_Init();
    MX_TIM3_Init();

    /* Initialize interrupts */
    MX_NVIC_Init();
    /* USER CODE BEGIN 2 */
    init_LCD();
    HAL_ADCEx_Calibration_Start(&hadc);
    // PWM setup
    uint32_t CCR = 0;
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3); // Start PWM on TIM3
Channel 3
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        // Toggle LED0
        HAL_GPIO_TogglePin(GPIOB, LED7_Pin);

        // ADC to LCD; TODO: Read POT1 value and write to LCD
        adc_val = pollADC();
        uint8_t bits[] = {-1, -1, -1, -1}, str_len = 0;
        //int to array of bits
        while (adc_val != 0) {
            bits[str_len++] = adc_val % 10;
            adc_val /= 10;
        }
        //initialise string with 0 as the default
        char str[5] = "0\0\0\0\0";
        //reverse bit array and convert to string
        for (uint8_t i = 0; i < str_len; i++) str[i] = '0' +
bits[str_len - i - 1];
        writeLCD(str);

        // Update PWM value; TODO: Get CRR
        CCR = ADCtoCCR(pollADC());
        __HAL_TIM_SetCompare(&htim3, TIM_CHANNEL_3, CCR);

        // Wait for delay ms
        HAL_Delay (delay_t);
```

```
        /* USER CODE END WHILE */

        /* USER CODE BEGIN 3 */
    }
    /* 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_RCC_HSI_Enable();

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

    }
    LL_RCC_HSI_SetCalibTrimming(16);
    LL_RCC_HSI14_Enable();

    /* Wait till HSI14 is ready */
    while(LL_RCC_HSI14_IsReady() != 1)
    {

    }
    LL_RCC_HSI14_SetCalibTrimming(16);
    LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
    LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_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_SetSystemCoreClock(8000000);

    /* Update the time base */
    if (HAL_InitTick (TICK_INT_PRIORITY) != HAL_OK)
    {
        Error_Handler();
    }
    LL_RCC_HSI14_EnableADCControl();
}

/**
```

```
  * @brief NVIC Configuration.
  * @retval None
  */
static void MX_NVIC_Init(void)
{
    /* EXTI0_1_IRQn interrupt configuration */
    NVIC_SetPriority(EXTI0_1_IRQn, 0);
    NVIC_EnableIRQ(EXTI0_1_IRQn);
}

/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    // TODO: Add code to switch LED7 delay frequency
    if (GPIO_Pin == Button0_Pin) {
        if (HAL_GetTick() - previous_tick > 10) {
            if (delay_t == 500) {
                delay_t = 1000;
            } else {
                delay_t = 500;
            }
        }
        previous_tick = HAL_GetTick();
    }
}


// TODO: Complete the writeLCD function
void writeLCD(char *char_in){
    delay(3000);
    lcd_command(CLEAR);
    lcd_putstring(char_in);
}

// Get ADC value
uint32_t pollADC(void){
    // TODO: Complete function body to get ADC val
    HAL_ADC_Start(&hadc);
    HAL_ADC_PollForConversion(&hadc, 1000);
    return HAL_ADC_GetValue(&hadc);
}

// Calculate PWM CCR value
uint32_t ADCtoCCR(uint32_t adc_val){
    // TODO: Calculate CCR val using an appropriate equation
    return adc_val * 47999 / 4095;
}

void ADC1_COMP_IRQHandler(void)
{
    adc_val = HAL_ADC_GetValue(&hadc); // read adc value
    HAL_ADC_IRQHandler(&hadc); //Clear flags
}
/* 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 */
```