```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 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"
#include "adc.h"
#include "dma.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "led.h"
#include "interrupt.h"
#include "stdio.h"
#include "string.h"
#include "lcd.h"
#include "i2c_hal.h"
#include "seg.h"
#include "ds18b20.h"
#include "dht11.h"


/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
extern struct keys key[4];
extern char rx_data;
extern char rx_arry[50];
extern char rx_pointer;
extern uint PA1_freq,PA1_duty;

/* 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 */
char lcd_arry[50];
char lcd_view;
u16 ADC1_array[2];
u16 ADC2_array[3];
uint PA7_freq = 1000;
uint PA7_duty = 50;
uint PA7_autoreload,PA7_compare;
```

```c

char LCD_view2_REC_PA_flag;//0, PA4
int PA4_vol_array[150];
int PA5_vol_array[150];
__IO uint32_t key_uwTick;
//char PARA_X_value;
//char PARA_Y_value;

int PARA_X_value;
int PARA_Y_value;

char vol_measure_flag;
char pwm_mode_flag;//0:倍频        1: 分频
char clear_record_flag;

__IO uint32_t eeprom_uwTick;
int REC_PA4_N_value;
float REC_PA4_A_value,REC_PA4_T_value,REC_PA4_H_value;
int REC_PA5_N_value;
float REC_PA5_A_value,REC_PA5_T_value,REC_PA5_H_value;

char lcd_mode_flag;//0,正向; 1, 翻转
__IO uint32_t led_uwTick;
__IO uint32_t pwm_uwTick;

char led_num;

char test_0;
char test_1;


/* USER CODE END PV */

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

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void key_proc(void);
void rx_proc(void);
void lcd_proc(void);
void eeprom_proc(void);
void pwm_proc(void);
void vol_measure(void);
void led_proc(void);
/* 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_DMA_Init();
  MX_TIM6_Init();
  MX_USART1_UART_Init();
  MX_TIM8_Init();
  MX_TIM16_Init();
  MX_ADC1_Init();
  MX_ADC2_Init();
  MX_TIM2_Init();
  MX_TIM3_Init();
  MX_TIM17_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  led_disp(0x00);

  HAL_TIM_Base_Start_IT(&htim6);

  HAL_UART_Receive_IT(&huart1, (uint8_t *)&rx_data, 1);

  LCD_Init();
  LCD_Clear(Black);
  LCD_SetTextColor(White);
  LCD_SetBackColor(Black);

  I2CInit();
//
  HAL_TIM_IC_Start_IT(&htim16,TIM_CHANNEL_1);
  HAL_TIM_IC_Start_IT(&htim8,TIM_CHANNEL_1);
  HAL_TIM_IC_Start_IT(&htim8,TIM_CHANNEL_2);
  HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_2);

  HAL_ADC_Start_DMA(&hadc1, (uint32_t *)ADC1_array,2);

  HAL_ADC_Start_DMA(&hadc2, (uint32_t *)ADC2_array,3);

  HAL_TIM_PWM_Start(&htim17,TIM_CHANNEL_1);

  led_disp(0x00);

  test_0 = eeprom_read(0);
  test_1 = eeprom_read(1);

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

    /* USER CODE BEGIN 3 */
     key_proc();
     rx_proc();
     lcd_proc();
     pwm_proc();
//   eeprom_proc();
     led_proc();
     vol_measure();
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV3;
  RCC_OscInitStruct.PLL.PLLN = 20;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}

/* USER CODE BEGIN 4 */
void led_proc(void)
{
    if(uwTick - led_uwTick < 100) return;
    led_uwTick = uwTick;

    if(pwm_mode_flag == 0)//倍频
        led_num |= 0x01;
    else led_num &= 0xfe;

    if(pwm_mode_flag == 1)//分频
        led_num |= 0x02;
    else led_num &= 0xfd;

    if(ADC2_array[1] > ADC2_array[2]*eeprom_read(1))
        led_num ^= 0x04;
    else led_num &= 0xfb;

    if(lcd_mode_flag == 0)//正向
        led_num |= 0x08;
    else led_num &= 0xf7;

    led_disp(led_num);

}
void key_proc(void)
{
    if(uwTick - key_uwTick < 50) return;
    key_uwTick = uwTick;

    for(int i=0;i<4;i++)
    {
        if(key[i].short_flag == 1 || key[i].long_flag == 1)
            LCD_Clear(Black);
    }

    if(key[0].short_flag == 1)
    {
        key[0].short_flag = 0;
        lcd_view++;
        if(lcd_view == 3) lcd_view = 0;
        if(lcd_view == 2) LCD_view2_REC_PA_flag = 0;

    }

    if(key[1].short_flag == 1)
    {
        key[1].short_flag = 0;
        PARA_X_value++;
        if(PARA_X_value == 5) PARA_X_value = 1;

        eeprom_write(1,PARA_X_value);//
    }

    if(key[2].short_flag == 1)
    {
        key[2].short_flag = 0;
        PARA_Y_value++;
        if(PARA_Y_value == 5) PARA_Y_value = 1;

        eeprom_write(0,PARA_Y_value);//
    }

    if(key[3].short_flag == 1)
    {
        key[3].short_flag = 0;
        if(lcd_view == 0)
        {
            vol_measure_flag = 1;
        }

        if(lcd_view == 1)
        {
            pwm_mode_flag = !pwm_mode_flag;
        }

        if(lcd_view == 2)
        {
            LCD_view2_REC_PA_flag = !LCD_view2_REC_PA_flag;
        }
    }

    if(key[3].long_flag == 1)
    {
        key[3].long_flag = 0;
        clear_record_flag = 1;
    }

}

void pwm_proc(void)
{
    if(uwTick - pwm_uwTick < 100) return;
    pwm_uwTick = uwTick;
    if(pwm_mode_flag == 0) //倍频
    {
        PA7_freq = PA1_freq*PARA_X_value;
    }
    else //分频
        PA7_freq = PA1_freq/PARA_X_value;

    PA7_autoreload = 1000000/PA7_freq;
    PA7_compare = PA7_autoreload*PA7_duty/100;
    __HAL_TIM_SetAutoreload(&htim17,PA7_autoreload);
    __HAL_TIM_SetCompare(&htim17,TIM_CHANNEL_1,PA7_compare);

}
//void eeprom_proc(void)
//{
//  if(uwTick - eeprom_uwTick < 100) return;
//  eeprom_uwTick = uwTick;
//  eeprom_write(0,PARA_Y_value);//
//  eeprom_write(1,PARA_X_value);//

//}
void vol_measure(void)
{
    int PA4_max_reg = 0;
    int PA4_min_reg = 4095;
    int PA4_avg_reg = 0;

    int PA5_max_reg = 0;
    int PA5_min_reg = 4095;
    int PA5_avg_reg = 0;

    int PA4_sum_reg,PA5_sum_reg;

    if(vol_measure_flag == 1)
    {
        vol_measure_flag = 0;
        if(LCD_view2_REC_PA_flag == 0)
        {
            PA4_vol_array[REC_PA4_N_value++] = ADC2_array[1];
            for(int i=0;i<REC_PA4_N_value;i++)
            {
                if(PA4_vol_array[i] > PA4_max_reg)
                    PA4_max_reg = PA4_vol_array[i];
                if(PA4_vol_array[i] < PA4_min_reg)
                    PA4_min_reg = PA4_vol_array[i];
                PA4_sum_reg += PA4_vol_array[i];
            }

            PA4_avg_reg = PA4_sum_reg/REC_PA4_N_value;

            REC_PA4_A_value = PA4_max_reg*3.3/4096;
            REC_PA4_T_value = PA4_min_reg*3.3/4096;
            REC_PA4_H_value = PA4_avg_reg*3.3/4096;

        }
        else if(LCD_view2_REC_PA_flag == 1)
        {
                         [               ++] =           [2];
            for(int i=0;i<REC_PA5_N_value;i++)
            {
                if(PA5_vol_array[i] > PA5_max_reg)
                    PA5_max_reg = PA5_vol_array[i];
                if(PA5_vol_array[i] < PA5_min_reg)
                    PA5_min_reg = PA5_vol_array[i];
                PA5_sum_reg += PA5_vol_array[i];
            }

            PA5_avg_reg = PA5_sum_reg/REC_PA5_N_value;

            REC_PA5_A_value = PA5_max_reg*3.3/4096;
            REC_PA5_T_value = PA5_min_reg*3.3/4096;
            REC_PA5_H_value = PA5_avg_reg*3.3/4096;
        }

    }

    if(clear_record_flag == 1)
    {
        clear_record_flag = 0;
        if(LCD_view2_REC_PA_flag == 0)
        {
            REC_PA4_N_value = 0;
            REC_PA4_A_value = 0;
            REC_PA4_T_value = 0;
            REC_PA4_H_value = 0;
        }

        if(LCD_view2_REC_PA_flag == 1)
        {
            REC_PA5_N_value = 0;
            REC_PA5_A_value = 0;
            REC_PA5_T_value = 0;
            REC_PA5_H_value = 0;
        }
    }

}
void lcd_proc(void)
{

    if(lcd_view == 0)
    {
        sprintf(lcd_arry,"DATA");
        LCD_DisplayStringLine(Line1, (u8 *)lcd_arry);

        sprintf(lcd_arry,"PA4=%-4.2f",ADC2_array[1]*3.3/4096);
        LCD_DisplayStringLine(Line3, (u8 *)lcd_arry);

        sprintf(lcd_arry,"PA5=%-4.2f",ADC2_array[2]*3.3/4096);
        LCD_DisplayStringLine(Line4, (u8 *)lcd_arry);

        sprintf(lcd_arry,"PA1=%-6d",PA1_freq);
        LCD_DisplayStringLine(Line5, (u8 *)lcd_arry);
    }
    else if(lcd_view == 1)
    {
        sprintf(lcd_arry,"PARA");
        LCD_DisplayStringLine(Line1, (u8 *)lcd_arry);

        sprintf(lcd_arry,"X=%d",eeprom_read(1));
        LCD_DisplayStringLine(Line3, (u8 *)lcd_arry);

        sprintf(lcd_arry,"Y=%d",eeprom_read(0));
        LCD_DisplayStringLine(Line4, (u8 *)lcd_arry);
    }
    else if(lcd_view == 2)
    {
        if(LCD_view2_REC_PA_flag == 0)
        {
            sprintf(lcd_arry,"REC-PA4");
            LCD_DisplayStringLine(Line1, (u8 *)lcd_arry);

            sprintf(lcd_arry,"N=%-4d",REC_PA4_N_value);
            LCD_DisplayStringLine(Line3, (u8 *)lcd_arry);

            sprintf(lcd_arry,"A=%-4.2f",REC_PA4_A_value);
            LCD_DisplayStringLine(Line4, (u8 *)lcd_arry);

            sprintf(lcd_arry,"T=%-4.2f",REC_PA4_T_value);
            LCD_DisplayStringLine(Line5, (u8 *)lcd_arry);

            sprintf(lcd_arry,"H=%-4.2f",REC_PA4_H_value);
            LCD_DisplayStringLine(Line6, (u8 *)lcd_arry);
        }
        else if(LCD_view2_REC_PA_flag == 1)
        {
            sprintf(lcd_arry,"REC-PA5");
            LCD_DisplayStringLine(Line1, (u8 *)lcd_arry);

            sprintf(lcd_arry,"N=%-4d",REC_PA5_N_value);
            LCD_DisplayStringLine(Line3,  (u8 *)lcd_arry);

            sprintf(lcd_arry,"A=%-4.2f",REC_PA5_A_value);
            LCD_DisplayStringLine(Line4,  (u8 *)lcd_arry);

            sprintf(lcd_arry,"T=%-4.2f",REC_PA5_T_value);
            LCD_DisplayStringLine(Line5,  (u8 *)lcd_arry);

            sprintf(lcd_arry,"H=%-4.2f",REC_PA5_H_value);
            LCD_DisplayStringLine(Line6,  (u8 *)lcd_arry);
        }

    }



}
void rx_proc(void)
{
    //判断数据是否接受完毕
    if(rx_pointer!= 0)
    {
        int temp = rx_pointer;
        //接收一次数据需要9个Bit
        HAL_Delay(1);//如果数据没有接受完毕，那么在这1ms内一定会发生中断，rx_pointer一定会变化
        //之所以1ms内一定会发生中断是因为最小的时间是每个字节接收结束到下个字节开始接收的这段时间
        //显然这段时间小于1ms，1ms能处理9bit，间隔时间一定小于9bit，
        if(temp == rx_pointer)
        {
            //串口接收处理部分
            if(strcmp(rx_arry,"X") == 0)
                printf("X:%d\n",eeprom_read(1));
            else if(strcmp(rx_arry,"Y") == 0)
                printf("Y:%d\n",eeprom_read(0));
            else if(strcmp(rx_arry,"PA1") == 0)
                printf("PA1:%d\n",PA1_freq);
            else if(strcmp(rx_arry,"PA4") == 0)
                printf("PA4:%.2f\n",ADC2_array[1]*3.3/4096);
            else if(strcmp(rx_arry,"PA5") == 0)
                printf("PA5:%.2f\n",ADC2_array[2]*3.3/4096);
            else if(strcmp(rx_arry,"#") == 0)
                lcd_mode_flag = !lcd_mode_flag;

            if(lcd_mode_flag == 0)
            {
                //上——>下，左——>右
                LCD_WriteReg(R1, 0x0000);   // set SS and SM bit                  //0x0100
                LCD_WriteReg(R96, 0x2700);  // Gate Scan Line                  0xA700
                LCD_Clear(Black);
            }
            else
            {
                //下——>上，右——>左
                LCD_WriteReg(R1, 0x0100);   // set SS and SM bit                  //0x0100
                LCD_WriteReg(R96, 0xA700);  // Gate Scan Line                  0xA700
                LCD_Clear(Black);
            }

            printf("%s\n",rx_arry);

            rx_pointer=0;memset(rx_arry,0,50);
        }
    }
}
int fputc(int ch, FILE *f)
{
  /* Your implementation of fputc(). */
    HAL_UART_Transmit(&huart1, (const uint8_t *)&ch,  1,  50);
    return ch;
}
```

```c


/* 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 */

```