```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "led.h"
#include "interrupt.h"
#include "string.h"

/* USER CODE END Includes */

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

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define PA1_FREQ 1000000
#define PA7_FREQ 1000000
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
extern uchar rx_arry[50];
extern uchar rx_data;
extern uchar rx_pointer;
extern struct keys key[4];


/* USER CODE END PM */

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

/* USER CODE BEGIN PV */
char lcd_arry[50];
uchar lcd_view;
uint pa1_freq = 1000;
uint pa1_duty = 10;
uint pa7_freq = 1000;
uint pa7_duty = 10;
uchar ctrl_mode = 0;
char led_num;
__IO uint32_t led_uwTick;
__IO uint32_t pwm_uwTick;
uint pa1_autoreload,pa1_compare;
uint pa7_autoreload,pa7_compare;

```

```c
/* 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 rx_proc(void);
void lcd_proc(void);
void key_proc(void);
void led_proc(void);
void pwm_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_TIM2_Init();
  MX_TIM6_Init();
  MX_TIM17_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */

    LCD_Init();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */

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

    HAL_TIM_Base_Start_IT(&htim6);//开启按键中断
    HAL_TIM_PWM_Start(&htim17,TIM_CHANNEL_1);//开启PWM输出
    HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_2);//开启PWM输出

    HAL_UART_Receive_IT(&huart1, (uint8_t *)&rx_data, 1);//开启串口接收中断

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

    /* USER CODE BEGIN 3 */

    if(rx_pointer!=0)
    {
        char temp = rx_pointer;
        HAL_Delay(1);
        if(temp == rx_pointer)
            rx_proc();
    }

    key_proc();
    lcd_proc();
    led_proc();
    pwm_proc();

    }
  /* 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_3) != HAL_OK)
  {
    Error_Handler();
  }
}

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

    pa1_autoreload = PA1_FREQ/pa1_freq;
    pa1_compare = pa1_duty*pa1_autoreload/100;
    pa7_autoreload = PA7_FREQ/pa7_freq;
    pa7_compare = pa7_duty*pa7_autoreload/100;

    __HAL_TIM_SetAutoreload(&htim2,pa1_autoreload);
    __HAL_TIM_SetCompare(&htim2,TIM_CHANNEL_2,pa1_compare);

    __HAL_TIM_SetAutoreload(&htim17,pa7_autoreload);
    __HAL_TIM_SetCompare(&htim17,TIM_CHANNEL_1,pa7_compare);

}
void led_proc()
{
    if(uwTick-led_uwTick<100)
        return;
    led_uwTick = uwTick;


    if(ctrl_mode == 0)
        led_num = led_num|0x04;//LED3亮
    else led_num = led_num&0xfb;//LED3灭

    if(pa1_freq>pa7_freq)                       //优先判断
        led_num = led_num^0x01;//LED1翻转闪烁
    else if(pa1_freq<pa7_freq)                  //条件级高
     led_num = led_num^0x02;//LED2翻转闪烁
    else if(lcd_view == 0)
    {
        led_num = led_num|0x01;//LED1亮
        led_num = led_num&0xfd;//LED2灭
    }

    else if(lcd_view == 1)
    {
        led_num = led_num|0x02;//LED2亮
        led_num = led_num&0xfe;//LED1灭
    }



    led_disp(led_num);


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

        if(lcd_view == 0)
        {
            pa1_freq = pa1_freq + 1000;
            if(pa1_freq > 10000)
                pa1_freq = 1000;
        }
        else
        {
            pa7_freq = pa7_freq + 1000;
            if(pa7_freq > 10000)
                pa7_freq = 1000;
        }
    }

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

        if(lcd_view == 0)
        {
            pa1_duty = pa1_duty + 10;
            if(pa1_duty > 90)
                pa1_duty = 10;
        }
        else
        {
            pa7_duty = pa7_duty + 10;
            if(pa7_duty > 90)
                pa7_duty = 10;
        }
    }

    if(key[2].short_flag == 1)
    {
        key[2].short_flag = 0;
        if(ctrl_mode == 0)
            lcd_view = !lcd_view;
    }

    if(key[3].short_flag == 1)
    {
        key[3].short_flag = 0;
        ctrl_mode = !ctrl_mode;
    }

}
void lcd_proc()
{
    if(lcd_view == 0)//PA1数据界面
    {
        sprintf(lcd_arry,"     PA1");
        LCD_DisplayStringLine(Line2, (u8 *)lcd_arry);
        sprintf(lcd_arry,"    F:%-dHZ",pa1_freq);
        LCD_DisplayStringLine(Line3, (u8 *)lcd_arry);
        sprintf(lcd_arry,"    D:%-2d%%",pa1_duty);
        LCD_DisplayStringLine(Line4, (u8 *)lcd_arry);
    }
    else
    {
        sprintf(lcd_arry,"     PA7");
        LCD_DisplayStringLine(Line2, (u8 *)lcd_arry);
        sprintf(lcd_arry,"    F:%-dHZ",pa7_freq);
        LCD_DisplayStringLine(Line3, (u8 *)lcd_arry);
        sprintf(lcd_arry,"    D:%-2d%%",pa7_duty);
        LCD_DisplayStringLine(Line4, (u8 *)lcd_arry);
    }
}
void rx_proc()
{
    if(ctrl_mode == 1)
    {
        if(rx_pointer == 1)
        {
            if(rx_arry[0] == '@')
                lcd_view = 0;
            else if(rx_arry[0] == '#')
                lcd_view = 1;
            else printf("ERROR\n");
        }
        else printf("ERROR\n");

    }
    else printf("KEY_CONTROL\n");

    rx_pointer = 0;memset(rx_arry,0,50);
}
int fputc(int ch, FILE *f)
{
    HAL_UART_Transmit(&huart1, (const uint8_t *)&ch, 1, 20);
  return ch;
}
/* 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 */

  /* 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,
       tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

```