```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 "adc.h"
#include "dma.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "lcd.h"
#include "led.h"
#include "interrupt.h"
#include "stdio.h"
#include "string.h"
/* 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 */
extern char rx_arry[50];
extern char rx_data;
extern char rx_pointer;
extern struct keys key[4];
/* USER CODE END PM */

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

/* USER CODE BEGIN PV */
char led_num;
uint16_t R37_arry[1];
uint16_t R38_arry[1];
float R37_vol,R38_vol;
char lcd_arry[50];
char lcd_view;
char R37_check_flag,R38_check_flag;
float SR37_min = 1.2;
float SR37_max = 2.2;
float SR38_min = 1.4;
float SR38_max = 3.0;
uchar standard_state;
uchar change_state;
uchar clear_flag;
float R37_ok_rate;
```

```c
uint R37_check_num,R37_ok_num;
float R38_ok_rate;
uint R38_check_num,R38_ok_num;
uint R37_ok_flag,R38_ok_flag;
uchar SR37_min_clear_flag;
uchar SR37_max_clear_flag;
uchar SR38_min_clear_flag;
uchar SR38_max_clear_flag;
__IO uint32_t led_uwTick;;
/* 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 key_proc();
void led_proc();
void lcd_proc();
void rate_proc();
/* 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_ADC1_Init();
  MX_ADC2_Init();
  MX_TIM6_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_ADC_Start_DMA(&hadc1, (uint32_t *)R38_arry,1);
    HAL_ADC_Start_DMA(&hadc2, (uint32_t *)R37_arry,1);

    HAL_TIM_Base_Start_IT(&htim6);

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

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

    /* USER CODE BEGIN 3 */


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

        key_proc();
        led_proc();
        lcd_proc();
        rate_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_2) != HAL_OK)
  {
    Error_Handler();
  }
}

/* USER CODE BEGIN 4 */
void rate_proc()
{
    R37_vol = R37_arry[0]*3.3/4096;
    R38_vol = R38_arry[0]*3.3/4096;

    if(R37_check_flag == 1)
    {
        R37_check_flag = 0;
        if(R37_vol>=SR37_min&&R37_vol<=SR37_max)
        {
            R37_ok_num++;
            R37_ok_flag = 1;
            led_uwTick = uwTick;
        }

        R37_check_num++;
        R37_ok_rate = R37_ok_num*100/R37_check_num;
    }

    if(R38_check_flag == 1)
    {
        R38_check_flag = 0;
        if(R38_vol>=SR38_min&&R38_vol<=SR38_max)
        {
            R38_ok_num++;
            R38_ok_flag = 1;
            led_uwTick = uwTick;
        }
        R38_check_num++;
        R38_ok_rate = R38_ok_num*100/R38_check_num;
    }

    if(clear_flag == 1)
    {
        clear_flag = 0;
        R37_ok_rate = 0;
        R38_ok_rate = 0;
    }

    if(SR37_max_clear_flag || SR37_min_clear_flag)
    {
        SR37_max_clear_flag = 0;
        SR37_min_clear_flag = 0;
        R37_ok_rate = 0;
    }

    if(SR38_max_clear_flag || SR38_min_clear_flag)
    {
        SR38_max_clear_flag = 0;
        SR38_min_clear_flag = 0;
        R38_ok_rate = 0;
    }
}
void lcd_proc()
{
    if(lcd_view == 0)
    {
        sprintf(lcd_arry,"       GOODS");
        LCD_DisplayStringLine(Line1,(u8 *)lcd_arry);
        sprintf(lcd_arry,"     R37:%-4.2fV",R37_vol);
        LCD_DisplayStringLine(Line3,(u8 *)lcd_arry);
        sprintf(lcd_arry,"     R38:%-4.2fV",R38_vol);
        LCD_DisplayStringLine(Line4,(u8 *)lcd_arry);
    }

    if(lcd_view == 1)
    {
        sprintf(lcd_arry,"      STANDARD");
        LCD_DisplayStringLine(Line1,(u8 *)lcd_arry);
        sprintf(lcd_arry,"    SR37:%-3.1f-%-3.1f",SR37_min,SR37_max);
        LCD_DisplayStringLine(Line3,(u8 *)lcd_arry);
        sprintf(lcd_arry,"    SR38:%-3.1f-%-3.1f",SR38_min,SR38_max);
        LCD_DisplayStringLine(Line4,(u8 *)lcd_arry);
    }

    if(lcd_view == 2)
    {
        sprintf(lcd_arry,"        PASS");
        LCD_DisplayStringLine(Line1,(u8 *)lcd_arry);
        sprintf(lcd_arry,"     PR37:%-4.1f%%",R37_ok_rate);
        LCD_DisplayStringLine(Line3,(u8 *)lcd_arry);
        sprintf(lcd_arry,"     PR38:%-4.1f%%",R38_ok_rate);
        LCD_DisplayStringLine(Line4,(u8 *)lcd_arry);
    }
}
void led_proc()
{
    if(R37_ok_flag == 1)
    {
        led_num = led_num|0x01;
        if(uwTick-led_uwTick>1000)
        {
            R37_ok_flag = 0;
            led_num = led_num&0x0fe;
        }
    }
    else led_num = led_num&0x0fe;

    if(R38_ok_flag == 1)
    {
        led_num = led_num|0x02;

        if(uwTick-led_uwTick>1000)
        {
            R38_ok_flag = 0;
            led_num = led_num&0x0fd;
        }
    }

    if(lcd_view == 0)
        led_num = led_num|0x04;//3亮
    else led_num = led_num&0xfb;//3灭

    if(lcd_view == 1)
        led_num = led_num|0x08;//4亮
    else led_num = led_num&0xf7;//4灭

    if(lcd_view == 2)
        led_num = led_num|0x10;//5亮
    else led_num = led_num&0xef;//5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;
        lcd_view++;
        if(lcd_view>2)
            lcd_view = 0;
    }
    if(key[1].short_flag == 1)
    {
```

```
		key[1].short_flag = 0;
		if(lcd_view == 0)//产品参数界面
		{
			R37_check_flag = 1;//R37合格率检测
		}
		if(lcd_view == 1)//标准设置界面
		{
			standard_state++;
			if(standard_state>3)
				standard_state = 0;
		}
	}

	if(key[2].short_flag == 1)
	{
		key[2].short_flag = 0;
		if(lcd_view == 0)
		{
			R38_check_flag = 1;//R38合格率检测
		}
		if(lcd_view == 1)
		{
			switch(standard_state)
			{
				case 0:
					SR37_max+=0.2;
					SR37_max_clear_flag = 1;
					if(SR37_max>3.0) SR37_max = 2.2;
					break;
				case 1:
					SR37_min+=0.2;
					SR37_min_clear_flag = 1;
					if(SR37_min>2.0) SR37_min = 1.2;
					break;
				case 2:
					SR38_max+=0.2;
					SR38_max_clear_flag = 1;
					if(SR38_max>3.0) SR38_max = 2.2;
					break;
				case 3:
					SR38_min+=0.2;
					SR38_min_clear_flag = 1;
					if(SR38_min>2.0) SR38_min = 1.2;
					break;
			}

		}
	}


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

		if(lcd_view == 0)
		{
			clear_flag = 1;//清零合格率
		}

		if(lcd_view == 1)
		{
			switch(standard_state)
			{
				case 0:
					SR37_max-=0.2;
					SR37_max_clear_flag = 1;
					if(SR37_max<2.2) SR37_max = 3.0;
					break;
				case 1:
					SR37_min-=0.2;
					SR37_min_clear_flag = 1;
```

```c
                    if(SR37_min<1.2) SR37_min = 2.0;
                    break;
                case 2:
                    SR38_max-=0.2;
                    SR38_max_clear_flag = 1;
                    if(SR38_max<2.2) SR38_max = 3.0;
                    break;
                case 3:
                    SR38_min-=0.2;
                    SR38_min_clear_flag = 1;
                    if(SR38_min<1.2) SR38_min = 2.0;
                    break;
            }
        }
    }

}
void rx_proc()
{
    if(rx_pointer == 3)
    {
        if(strcmp(rx_arry,"R37") == 0)
            printf("R37:%d,%d,%.1f%%",R37_check_num,R37_ok_num,R37_ok_rate);
        if(strcmp(rx_arry,"R38") == 0)
            printf("R38:%d,%d,%.1f%%",R38_check_num,R38_ok_num,R38_ok_rate);
    }
    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 */
```