```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"


/* 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 = 70;
uint PA7_autoreload,PA7_compare;
```

```
72
73  //温度
74  float temp_float;
75  __IO uint32_t  ds18b20_uwTick;
76
77
78  char PARA_change_state = 0;//0:FH   1:AH    2:TH
79  char FSET_change_state = 0;//0:FH   1:VP    2:TT
80  char record_flag;//开始记录标志位
81  __IO uint32_t  key_uwTick;
82
83  int R37_adc_record_array[100];//ADC记录数组
84  int PA1_freq_record_array[100];//频率记录数组
85  int PA1_duty_record_array[100];//占空比记录数组
86
87  char RECD_clear_flag;//统计参数清零
88
89  int PARA_FH_value = 2000;
90  float PARA_AH_value = 3.0;
91  int PARA_TH_value = 30;
92
93  int FSET_FP_value = 1;
94  float FSET_VP_value = 0.9;
95  int FSET_TT_value = 6;
96
97  char FSET_voltage_flag;//开始记录电压标志位
98  char FSET_pulse_flag;//开始频率和占空比标志位
99  __IO uint32_t FSET_voltage_flag_uwTick;
100 __IO uint32_t FSET_pulse_flag_uwTick;
101 char FSET_finish_flag;//记录完毕标志位
102
103 char Reset_flag;//复位标志位
104 __IO uint32_t  led_uwTick;
105
106 char led_num;
107
108 char F_alarm_flag;//频率报警信号
109 char V_alarm_flag;//电压报警信号
110 char T_alarm_flag;//温度报警信号
111
112 char RECD_FN_value;
113 char RECD_AN_value;
114 char RECD_TN_value;
115 __IO uint32_t  signal_record_uwTick;
116 __IO uint32_t  signal_review_uwTick;
117 __IO uint32_t  TT_time_uwTick;
118
119 int FH_reg,TH_reg;//上限参数寄存变量
120 float AH_reg;
121
122
123 //char maopao_array[10] = {0,6,8,2,9,1,7,3,5,4};
124 char maopao_array[50];
125
126 /* USER CODE END PV */
127
128 /* Private function prototypes -----------------------------------------------*/
129 void SystemClock_Config(void);
130 /* USER CODE BEGIN PFP */
131
132 /* USER CODE END PFP */
133
134 /* Private user code ---------------------------------------------------------*/
135 /* USER CODE BEGIN 0 */
136 void key_proc(void);
137 void rx_proc(void);
138 void lcd_proc(void);
139 void pwm_proc(void);
140 void ds18b20_proc(void);
141 void led_proc(void);
142 void signal_record(void);//信号记录函数
```

```
143   void signal_review(void);//信号回放函数
144   void alarm_proc(void);//报警函数
145   void maopao_sort(int length,char maopao_array[]);
146
147   /* USER CODE END 0 */
148
149   /**
150     * @brief  The application entry point.
151     * @retval int
152     */
153   int main(void)
154   {
155     /* USER CODE BEGIN 1 */
156
157     /* USER CODE END 1 */
158
159     /* MCU Configuration--------------------------------------------------------*/
160
161     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
162     HAL_Init();
163
164     /* USER CODE BEGIN Init */
165
166     /* USER CODE END Init */
167
168     /* Configure the system clock */
169     SystemClock_Config();
170
171     /* USER CODE BEGIN SysInit */
172
173     /* USER CODE END SysInit */
174
175     /* Initialize all configured peripherals */
176     MX_GPIO_Init();
177     MX_DMA_Init();
178     MX_TIM6_Init();
179     MX_USART1_UART_Init();
180     MX_TIM8_Init();
181     MX_TIM16_Init();
182     MX_ADC1_Init();
183     MX_ADC2_Init();
184     MX_TIM2_Init();
185     MX_TIM3_Init();
186     MX_TIM17_Init();
187     /* USER CODE BEGIN 2 */
188
189     /* USER CODE END 2 */
190
191     /* Infinite loop */
192     /* USER CODE BEGIN WHILE */
193     led_disp(0x00);
194
195     HAL_TIM_Base_Start_IT(&htim6);
196
197     HAL_UART_Receive_IT(&huart1,(uint8_t *)&rx_data,1);
198
199     LCD_Init();
200     LCD_Clear(Black);
201     LCD_SetTextColor(White);
202     LCD_SetBackColor(Black);
203
204     I2CInit();
205   //
206     HAL_TIM_IC_Start_IT(&htim16,TIM_CHANNEL_1);//按键扫描定时器中断
207     HAL_TIM_IC_Start_IT(&htim8,TIM_CHANNEL_1);
208     HAL_TIM_IC_Start_IT(&htim8,TIM_CHANNEL_2);
209     HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_2);//测量PA1的频率和占空比
210   //  HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_3);
211
212     HAL_ADC_Start_DMA(&hadc1,(uint32_t *)ADC1_array,2);
213     HAL_ADC_Start_DMA(&hadc2,(uint32_t *)ADC2_array,3);
```

```
214
215   HAL_TIM_PWM_Start(&htim17,TIM_CHANNEL_1);//PA7--PWM输出
216
217   led_disp(0x00);
218   ds18b20_init_x();
219   while((int)ds18b20_read() == 85) {}
220
221   while (1)
222   {
223     /* USER CODE END WHILE */
224
225     /* USER CODE BEGIN 3 */
226      key_proc();
227      rx_proc();
228      lcd_proc();
229      ds18b20_proc();
230      pwm_proc();
231      led_proc();
232      signal_record();
233      signal_review();
234      alarm_proc();
235   }
236   /* USER CODE END 3 */
237 }
238
239 /**
240   * @brief System Clock Configuration
241   * @retval None
242   */
243 void SystemClock_Config(void)
244 {
245   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
246   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
247
248   /** Configure the main internal regulator output voltage
249   */
250   HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);
251
252   /** Initializes the RCC Oscillators according to the specified parameters
253   * in the RCC OscInitTypeDef structure.
254   */
255   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
256   RCC_OscInitStruct.HSEState = RCC_HSE_ON;
257   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
258   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
259   RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV3;
260   RCC_OscInitStruct.PLL.PLLN = 20;
261   RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
262   RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
263   RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
264   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
265   {
266     Error_Handler();
267   }
268
269   /** Initializes the CPU, AHB and APB buses clocks
270   */
271   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
272                               |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
273   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
274   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
275   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
276   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
277
278   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
279   {
280     Error_Handler();
281   }
282 }
283
284 /* USER CODE BEGIN 4 */
```

```c
void maopao_sort(int length,char maopao_array[])
{
        char temp;//交换两个变量的值需要一个中间变量
        //如果length为10
        //总共比较：length-1次
        //i表示每轮比较结束元素的索引+1，
        //第一轮：8+1
        //第一轮：7+1
        //第一轮：6+1
        //...
        //第一轮：0+1
        for(int i=length-1;i>0;i--)
        {
                //每次都从第一个元素开始相邻比较，结束元素索引不同
                //0----8，<9:第0个元素和第1个元素比较，...，第8个元素和第9个元素比较
                //0----0，<1: 第0个元素和第1个元素比较
                for(int j=0;j<i;j++)
                {
                        if(maopao_array[j]> maopao_array[j+1])
                        {
                                temp = maopao_array[j+1];
                                maopao_array[j+1]= maopao_array[j];
                                maopao_array[j] = temp;
                        }

                }
        }

        //打印查看信息
        for(int i=0;i<length;i++)
        {
                printf("%c\n",maopao_array[i]);
        }
}
void alarm_proc(void)
{
    //从报警参数界面退出时，新的 FH、AH 和TH 参数生效
    if(lcd_view != 1) //非报警参数界面时更新参数
    {
        FH_reg = PARA_FH_value;
        AH_reg = PARA_AH_value;
        TH_reg = PARA_TH_value;
    }

    //大于上限，只累加一次
    //F_alarm_flag为0表示频率值低于上限或者初始状态
    //+10和-10是设置一个滞环，减小波动的影响
    //上大于上限+10，且当前频率值低于下限，则累加一次，同时标志位拉高说明此时频率值大于上限值
    if(PA1_freq > FH_reg+10 && (F_alarm_flag == 0))
    {
        F_alarm_flag = 1;
        RECD_FN_value++;
    }
    else if(PA1_freq <= FH_reg-10)
        F_alarm_flag = 0;

    //对电压也是同样的逻辑
    //但是电压是浮点数，不能判等
    if(ADC2_array[0]*3.3/4096 > AH_reg && (V_alarm_flag == 0))
    {
        V_alarm_flag = 1;
        RECD_AN_value++;
    }
    else if(ADC2_array[0]*3.3/4096  <= AH_reg)
        V_alarm_flag = 0;

    //温度带有小数，而上限参数是整型，所以放大10倍，小数也进行比较
    if((int)(temp_float*10) > (TH_reg*10) && (T_alarm_flag == 0))
    {
        T_alarm_flag = 1;
        RECD_TN_value++;
```

```
356         }
357         else if((int)(temp_float*10) <= (TH_reg*10))
358             T_alarm_flag = 0;
359 
360         //统计值清零
361         if(RECD_clear_flag)
362         {
363             RECD_FN_value = 0;
364             RECD_AN_value = 0;
365             RECD_TN_value = 0;
366             RECD_clear_flag = 0;//统计值清零标志拉低
367         }
368 
369         //复位，设备回到初始状态
370         if(Reset_flag)
371         {
372             Reset_flag = 0;
373 
374             //统计值复位
375             RECD_FN_value = 0;
376             RECD_AN_value = 0;
377             RECD_TN_value = 0;
378 
379             //上限值复位
380             PARA_FH_value = 2000;
381             PARA_AH_value = 3.0;
382             PARA_TH_value = 30;
383 
384             //记录值复位
385             FSET_FP_value = 1;
386             FSET_VP_value = 0.9;
387             FSET_TT_value = 6;
388 
389             //界面复位
390             lcd_view = 0;
391             PARA_change_state = 0;//0:FH    1:AH    2:TH
392             FSET_change_state = 0;//0:FH    1:VP    2:TT
393 
394         }
395 
396 }
397 void signal_record(void)
398 {
399     //每100ms记录一次数据到数组中
400     if(uwTick - signal_record_uwTick < 100) return;
401         signal_record_uwTick = uwTick;
402     static int i;
403 
404     //开始记录
405     if(record_flag == 1)
406     {
407         R37_adc_record_array[i] = ADC2_array[0];
408         PA1_freq_record_array[i] = PA1_freq;
409         PA1_duty_record_array[i] = PA1_duty;
410 
411         i++;
412 
413         //记录时间达到
414         if(uwTick - TT_time_uwTick > FSET_TT_value*1000)
415         {
416             i = 0;//数组索引复位
417             record_flag = 0;//标志位拉低
418             FSET_finish_flag = 1;//记录结束标志拉高
419         }
420     }
421 }
422 
423 void signal_review(void)
424 {
425     //每100ms回放一次数据
426     if(uwTick - signal_review_uwTick < 100) return;
```

```
427         signal_review_uwTick = uwTick;
428     static int j;
429 
430     //开始电压回放
431     if(FSET_voltage_flag)
432     {
433         PA7_freq = 1000;
434         PA7_duty = (int)((R37_adc_record_array[j]*3.3/4096-3.3)*90.0/(3.3-FSET_VP_value) + 100.0);
435         if((R37 adc record array[j]*3.3/4096) < FSET VP value)
436             PA7_duty = 10;
437         j++;
438 
439         //记录时间达到
440         if(uwTick - FSET_voltage_flag_uwTick > FSET_TT_value*1000)
441         {
442             FSET_voltage_flag = 0;//标志位拉低
443             j = 0;//数组索引复位
444             PA7 duty = 0;//回放结束，输出低电平
445         }
446     }
447 
448     //开始脉冲回放
449     if(FSET_pulse_flag)
450     {
451         PA7_freq = PA1_freq_record_array[j]/FSET_FP_value;
452         PA7_duty = PA1_duty_record_array[j];
453         j++;
454 
455         //记录时间达到
456         if(uwTick - FSET_pulse_flag_uwTick > FSET_TT_value*1000)
457         {
458             FSET_pulse_flag = 0;//标志位拉低
459             j = 0;//数组索引复位
460             PA7_duty = 0;//回放结束，输出低电平
461         }
462     }
463 }
464 void key_proc(void)
465 {
466     //每50ms执行一次按键处理
467     if((uwTick - key_uwTick < 50)) return;
468     key_uwTick = uwTick;
469 
470     for(int i=0;i<4;i++)
471     {
472         //正在记录数据，所有标志位清零，按键均无效
473         if(record_flag)
474         {
475             key[i].short_flag = 0;
476             key[i].long_flag = 0;
477         }//记录结束，按键按下，清一次屏
478         else if(key[i].short_flag == 1 || key[i].long_flag == 1)
479             LCD_Clear(Black);
480     }
481 
482 
483     if(key[0].short_flag == 1)
484     {
485         key[0].short_flag = 0;
486         lcd_view++;
487         //界面切换的时候，复位参数初始状态
488         if(lcd_view == 4) lcd_view = 0;
489         if(lcd_view == 1) PARA_change_state = 0;
490         if(lcd_view == 3) FSET_change_state = 0;
491     }
492 
493     if(key[1].short_flag == 1)
494     {
495         key[1].short_flag = 0;
496 
497         if(lcd_view == 0)
```

```
498         {
499             //开始记录
500             record_flag = 1;
501             TT_time_uwTick = uwTick;//开始计时
502             FSET_finish_flag = 0;//记录结束标志位拉低，初始状态没有记录数据的时候，不能执行回放
503         }
504 
505         if(lcd_view == 1)
506         {
507             //选中参数状态自加，超出范围，复位
508             PARA_change_state++;
509             if(PARA_change_state == 3) PARA_change_state = 0;
510         }
511 
512         //统计值清零
513         if(lcd_view == 2) RECD_clear_flag = 1;
514 
515         if(lcd view == 3)
516         {
517             FSET_change_state++;
518             if(FSET_change_state == 3) FSET_change_state = 0;
519         }
520     }
521 
522     if(key[2].short_flag == 1)
523     {
524         key[2].short_flag = 0;
525         if(lcd_view == 1)
526         {
527             //根据选中的参数执行自加或自减
528             //超出范围需要置位
529             switch(PARA_change_state)
530             {
531                 case 0:
532                     PARA_FH_value+=1000;
533                     if(PARA_FH_value >= 10000) PARA_FH_value = 10000;
534                     break;
535                 case 1:
536                     PARA_AH_value+=0.3f;
537                     if(PARA AH value >= 3.3f) PARA AH value = 3.3f;
538                     break;
539                 case 2:
540                     PARA_TH_value+=1;
541                     if(PARA_TH_value >= 80) PARA_TH_value = 80;
542                     break;
543             }
544         }
545 
546         if(lcd view == 3)
547         {
548             switch(FSET_change_state)
549             {
550                 case 0:
551                     FSET_FP_value+=1;
552                     if(FSET_FP_value >= 10) FSET_FP_value = 10;
553                     break;
554                 case 1:
555                     FSET_VP_value+=0.3f;
556                     if(FSET_VP_value >= 3.3f) FSET_VP_value = 3.3f;
557                     break;
558                 case 2:
559                     FSET_TT_value+=2;
560                     if(FSET_TT_value >= 10) FSET_TT_value = 10;
561                     break;
562             }
563         }
564 
565         if(lcd_view == 0)
566         {
567             //FSET_finish_flag为高说明已经记录过数据了，同时此刻并没有开始记录数据
568             //记录数据和回放不能同时进行
```

```
569             if(FSET_finish_flag)
570             {
571                 FSET_voltage_flag = 1;
572                 FSET_voltage_flag_uwTick = uwTick;//开始计时
573             }
574         }
575 
576     }
577 
578     if(key[3].short_flag == 1)
579     {
580         key[3].short_flag = 0;
581         if(lcd_view == 1)
582         {
583             switch(PARA_change_state)
584             {
585                 case 0:
586                     PARA_FH_value-=1000;
587                     if(PARA_FH_value <= 1000) PARA_FH_value = 1000;
588                     break;
589                 case 1:
590                     PARA_AH_value-=0.3f;
591                     if(PARA_AH_value <= 0.0f) PARA_AH_value = 0.0f;
592                     break;
593                 case 2:
594                     PARA_TH_value-=1;
595                     if(PARA_TH_value <= 0) PARA_TH_value = 0;
596                     break;
597             }
598         }
599 
600         if(lcd_view == 3)
601         {
602             switch(FSET_change_state)
603             {
604                 case 0:
605                     FSET_FP_value-=1;
606                     if(FSET_FP_value <= 1) FSET_FP_value = 1;
607                     break;
608                 case 1:
609                     FSET_VP_value-=0.3f;
610                     if(FSET_VP_value <= 0.0f) FSET_VP_value = 0.0f;
611                     break;
612                 case 2:
613                     FSET_TT_value-=2;
614                     if(FSET_TT_value <= 2) FSET_TT_value = 2;
615                     break;
616             }
617         }
618 
619         if(lcd_view == 0)
620         {
621             if(FSET_finish_flag)
622             {
623                 FSET_pulse_flag = 1;
624                 FSET_pulse_flag_uwTick = uwTick;
625             }
626 
627         }
628 
629     }
630 
631     //复位
632     if(key[2].long_flag && key[3].long_flag)
633     {
634         key[2].long_flag = 0;
635         key[3].long_flag = 0;
636         Reset_flag = 1;
637     }
638 
639 }
```

```
640  void led_proc(void)
641  {
642      //每0.1s执行一次，实现翻转功能
643      if(uwTick - led_uwTick < 100) return;
644          led_uwTick = uwTick;
645
646      //满足条件，翻转，否则清零
647      if(record_flag == 1)
648          led_num ^= 0x01;
649      else led_num &= 0xfe;
650
651      if(FSET_pulse_flag == 1)
652          led_num ^= 0x02;
653      else led_num &= 0xfd;
654
655      if(FSET_voltage_flag == 1)
656          led_num ^= 0x04;
657      else led_num &= 0xfb;
658
659      if(F_alarm_flag == 1)
660          led_num |= 0x08;
661      else led_num &= 0xf7;
662
663      if(V_alarm_flag == 1)
664          led_num |= 0x10;
665      else led_num &= 0xef;
666
667      if(T_alarm_flag == 1)
668          led_num |= 0x20;
669      else led_num &= 0xdf;
670
671      //显示LED
672      led_disp(led_num);
673
674  }
675  void ds18b20_proc(void)
676  {
677      //每0.1s读取一次温度值
678      if(uwTick - ds18b20_uwTick < 100)    return;
679      ds18b20_uwTick = uwTick;
680      temp_float = ds18b20_read();
681  }
682
683
684  void pwm_proc(void)
685  {
686      //配置PWM输出频率与占空比
687      PA7_autoreload = 1000000/PA7_freq;
688      PA7_compare = PA7_autoreload*PA7_duty/100;
689      __HAL_TIM_SetAutoreload(&htim17,PA7_autoreload);
690      __HAL_TIM_SetCompare(&htim17,TIM_CHANNEL_1,PA7_compare);
691  }
692
693  void lcd_proc(void)
694  {
695      //显示
696      if(lcd_view == 0)
697      {
698          //避免高亮的影响，保持原本的背景色
699          LCD_SetBackColor(Black);
700          sprintf(lcd_arry,"        DATA");
701          LCD_DisplayStringLine(Line1, (u8 *)lcd_arry);
702
703          sprintf(lcd_arry,"     F = %-6d",PA1_freq);
704          LCD_DisplayStringLine(Line3, (u8 *)lcd_arry);
705
706          sprintf(lcd_arry,"     D = %d%%",PA1_duty);
707          LCD_DisplayStringLine(Line4, (u8 *)lcd_arry);
708
709          sprintf(lcd_arry,"     A = %3.1f",ADC2_array[0]*3.3/4096);
710          LCD_DisplayStringLine(Line5, (u8 *)lcd_arry);
```

```
711
712         sprintf(lcd_arry,"     T = %4.1f",temp_float);
713         LCD_DisplayStringLine(Line6, (u8 *)lcd_arry);
714     }
715     else if(lcd_view == 1)
716     {
717         LCD_SetBackColor(Black);
718         sprintf(lcd_arry,"         PARA");
719         LCD DisplayStringLine(Line1, (u8 *)lcd arry);
720
721         //高亮选中的参数，否则保持原本的背景色
722         if(PARA_change_state == 0)  LCD_SetBackColor(Red);
723         else LCD_SetBackColor(Black);
724         sprintf(lcd_arry,"     FH = %-5d",PARA_FH_value);
725         LCD_DisplayStringLine(Line3, (u8 *)lcd_arry);
726
727         if(PARA_change_state == 1)  LCD_SetBackColor(Red);
728         else LCD_SetBackColor(Black);
729         sprintf(lcd_arry,"     AH = %3.1f",PARA_AH_value);
730         LCD_DisplayStringLine(Line4, (u8 *)lcd_arry);
731
732         if(PARA_change_state == 2)  LCD_SetBackColor(Red);
733         else LCD_SetBackColor(Black);
734         sprintf(lcd_arry,"     TH = %-2d",PARA_TH_value);
735         LCD_DisplayStringLine(Line5, (u8 *)lcd_arry);
736
737     }
738     else if(lcd_view == 2)
739     {
740         //避免高亮的影响，保持原本的背景色
741         LCD_SetBackColor(Black);
742         sprintf(lcd_arry,"         RECD");
743         LCD_DisplayStringLine(Line1, (u8 *)lcd_arry);
744
745         sprintf(lcd_arry,"     FN = %-2d",RECD_FN_value);
746         LCD_DisplayStringLine(Line3, (u8 *)lcd_arry);
747
748         sprintf(lcd_arry,"     AN = %-2d",RECD_AN_value);
749         LCD_DisplayStringLine(Line4, (u8 *)lcd_arry);
750
751         sprintf(lcd_arry,"     TN = %-2d",RECD_TN_value);
752         LCD_DisplayStringLine(Line5, (u8 *)lcd_arry);
753     }
754     else if(lcd_view == 3)
755     {
756         //避免高亮的影响，保持原本的背景色
757         LCD_SetBackColor(Black);
758         sprintf(lcd_arry,"         FSET");
759         LCD DisplayStringLine(Line1, (u8 *)lcd arry);
760
761         if(FSET_change_state == 0)  LCD_SetBackColor(Red);
762         else LCD_SetBackColor(Black);
763         sprintf(lcd_arry,"     FP = %-2d",FSET_FP_value);
764         LCD_DisplayStringLine(Line3, (u8 *)lcd_arry);
765
766         if(FSET_change_state == 1)  LCD_SetBackColor(Red);
767         else LCD_SetBackColor(Black);
768         sprintf(lcd_arry,"     VP = %3.1f",FSET_VP_value);
769         LCD_DisplayStringLine(Line4, (u8 *)lcd_arry);
770
771         if(FSET_change_state == 2)  LCD_SetBackColor(Red);
772         else LCD_SetBackColor(Black);
773         sprintf(lcd_arry,"     TT = %-2d",FSET_TT_value);
774         LCD_DisplayStringLine(Line5, (u8 *)lcd_arry);
775     }
776 }
777 void rx_proc(void)
778 {
779     //判断数据是否接受完毕
780     if(rx_pointer!= 0)
781     {
```

```
782         int temp = rx_pointer;
783         //接收一次数据需要9个Bit
784         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
785         //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
786         //显然这段时间小于1ms，1ms能处理9bit，间隔时间一定小于9bit，
787         if(temp == rx_pointer)
788         {
789             //串口接收处理部分
790             maopao sort(strlen(rx arry),rx arry);
791             printf("data_lenth:%d\n",strlen(rx_arry));
792 //          printf("%s\n",rx_arry);
793             led_disp(temp);
794             rx_pointer=0;memset(rx_arry,0,50);
795         }
796     }
797 }
798 
799 int fputc(int ch, FILE *f)
800 {
801   /* Your implementation of fputc(). */
802     HAL_UART_Transmit(&huart1, (const uint8_t *)&ch,  1, 50);
803     return ch;
804 }
805 
806 
807 /* USER CODE END 4 */
808 
809 /**
810   * @brief  This function is executed in case of error occurrence.
811   * @retval None
812   */
813 void Error_Handler(void)
814 {
815   /* USER CODE BEGIN Error_Handler_Debug */
816   /* User can add his own implementation to report the HAL error return state */
817   __disable_irq();
818   while (1)
819   {
820   }
821   /* USER CODE END Error Handler Debug */
822 }
823 
824 #ifdef  USE_FULL_ASSERT
825 /**
826   * @brief  Reports the name of the source file and the source line number
827   *         where the assert_param error has occurred.
828   * @param  file: pointer to the source file name
829   * @param  line: assert_param error line source number
830   * @retval None
831   */
832 void assert_failed(uint8_t *file, uint32_t line)
833 {
834   /* USER CODE BEGIN 6 */
835   /* User can add his own implementation to report the file name and line number,
836     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
837   /* USER CODE END 6 */
838 }
839 #endif /* USE_FULL_ASSERT */
840 
```