```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 "gpio.h"

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

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
extern uint pa7_freq,pa7_duty;
extern struct keys key[4];

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
void lcd_proc(void);
void led_proc(void);
void pwm_proc(void);
void key_proc(void);
void speed_proc(void);


/* USER CODE END PM */

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

/* USER CODE BEGIN PV */
char lcd_arry[50];
char lcd_view;
u16 adc2_arry[1];
float adc2_vol;
char pwm_mode,pwm_change_flag;
uint pa1_freq = 4000;
uint pa1_duty = 20;
uint pa1_autoreload,pa1_compare;
uint pa1_duty_lock;
float speed_value,H_max_speed,L_max_speed;
uint R_value = 1;
```

```
uint K_value = 1;
uint R_K_mode;
uint mode_change_sum;
char led_num;

__IO uint32_t pwm_mode_Tick = 0;//记时，记5s
__IO uint32_t speed_Tick = 0;//记时，2s后处理程序
__IO uint32_t speed_Tick_2 = 0;//定时,每隔0.1s执行函数
  IO uint32 t led Tick = 0;//闪烁
__IO uint32_t pwm_Tick = 0;//定时，每隔0.1s执行函数

/* 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 */

/* 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_ADC2_Init();
  MX_TIM2_Init();
  MX_TIM6_Init();
  MX_TIM17_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);
    HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_2);
    HAL_TIM_IC_Start_IT(&htim17,TIM_CHANNEL_1);
```

```
143     HAL_ADC_Start_DMA(&hadc2,(uint32_t *)adc2_arry,1);
144
145     while (1)
146     {
147     /* USER CODE END WHILE */
148
149     /* USER CODE BEGIN 3 */
150     lcd_proc();
151     led_proc();
152     pwm_proc();
153     key_proc();
154     speed_proc();
155
156     }
157   /* USER CODE END 3 */
158 }
159
160 /**
161   * @brief System Clock Configuration
162   * @retval None
163   */
164 void SystemClock_Config(void)
165 {
166   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
167   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
168
169   /** Configure the main internal regulator output voltage
170   */
171   HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);
172
173   /** Initializes the RCC Oscillators according to the specified parameters
174   * in the RCC_OscInitTypeDef structure.
175   */
176   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
177   RCC_OscInitStruct.HSEState = RCC_HSE_ON;
178   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
179   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
180   RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV3;
181   RCC_OscInitStruct.PLL.PLLN = 20;
182   RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
183   RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
184   RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
185   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
186   {
187     Error_Handler();
188   }
189
190   /** Initializes the CPU, AHB and APB buses clocks
191   */
192   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
193                               |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
194   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
195   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
196   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
197   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
198
199   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
200   {
201     Error_Handler();
202   }
203 }
204
205 /* USER CODE BEGIN 4 */
206 //uint pa1_freq,pa1_duty,pa1_duty_lock;
207 void pwm_proc()
208 {
209     if(uwTick - pwm_Tick < 100) //100ms
210         return;
211     pwm_Tick = uwTick;
212
213     if(pwm_change_flag == 1)
```

```
214         {
215             if(pwm_mode == 0) //低变高
216             {
217                 if(pa1_freq >=4000 && pa1_freq<8000)
218                     pa1_freq+= 100;
219             }
220             else
221             {
222                 if(pa1 freq >4000 && pa1 freq<=8000)
223                     pa1_freq-= 100;
224             }
225 
226         }
227 
228         adc2_vol = adc2_arry[0]*3.3/4096;
229         if(pa1_duty_lock == 0)
230         {
231             if(adc2 vol < 1)
232                 pa1_duty = 10;
233             else if(adc2_vol >3)
234                 pa1_duty = 85;
235             else pa1_duty = (uint)(37.5*adc2_vol-27.5);
236         }
237         else pa1_duty = pa1_duty;
238 
239 
240         pa1_autoreload = 1000000/pa1_freq;
241         pa1_compare = pa1_duty*pa1_autoreload/100;
242           HAL TIM SetAutoreload(&htim2,pa1 autoreload);
243         __HAL_TIM_SetCompare(&htim2,TIM_CHANNEL_2,pa1_compare);
244 
245 
246 }
247 void led_proc()
248 {
249     if(uwTick - led_Tick < 100)
250         return ;
251     led_Tick = uwTick;
252     led_disp(0x00);
253     // & | ^
254     if(lcd_view == 1)
255         led_num = led_num|0x01;
256 //  if(pwm_change_flag == 1)
257 //      led_num = led_num|0x01;
258 //  else led_num = led_num|0x01;
259 //
260 //  if(pa1_duty_lock == 1)
261 //      led_num = led_num|0x01;
262 //  else led num = led num|0x01;
263 //
264 }
265 
266 void speed_proc()
267 {
268     static float speed_max_L,speed_max_H;
269     static float speed_temp;
270     static float speed_max_reg;
271 
272     if(uwTick - speed_Tick_2 < 100)
273         return;
274     speed_Tick_2 = uwTick;//每0.1s执行一次
275 
276     speed_value = pa7_freq*2*3.14*R_value/(100*K_value);
277 
278     if((int)speed_value != (int)speed_temp)//发生变化开始计时
279         speed_Tick = uwTick;
280     if(uwTick -speed_Tick > 2000)//保持的时间持续了2秒
281         speed_max_reg = speed_value;
282 
283     speed_temp = speed_value;//每隔0.1s寄存数据
284 
```

```
285     if(pwm_mode == 1)//不同模式下比较大小更新对应的最大值
286     {
287         if(speed_max_reg > speed_max_H)//满足持续时间条件再进行大小;
288             speed_max_H = speed_max_reg;
289         H_max_speed = speed_max_H;
290     }
291 
292     else
293     {
294         if(speed_max_reg > speed_max_L)//满足条件再进行大小比较
295             speed_max_L = speed_max_reg;
296         L_max_speed = speed_max_L;
297     }
298 
299 }
300 void key_proc()
301 {
302     for(int i=0;i<4;i++)
303     {
304         if(key[i].short_flag == 1 || key[i].long_flag == 1)
305             LCD_Clear(Black);//按键按下则清屏
306     }
307 
308     if(key[0].short_flag == 1)
309     {
310         key[0].short_flag = 0;
311         lcd_view ++;//界面切换
312         if(lcd_view == 3) lcd_view = 0;
313     }
314 
315     if(key[1].short_flag == 1)
316     {
317         key[1].short_flag = 0;
318 
319         if(lcd_view == 0)
320             R_K_mode = 0;//从数据界面进入参数界面时默认R参数
321 
322         if(lcd_view == 0 &&((uwTick-pwm_mode_Tick>5000) || pwm_mode_Tick ==
    0))//第一次或者按下之后5秒后
323         {
324             pwm_mode_Tick = uwTick;
325             pwm_change_flag = 1;//切换过程标志
326             mode_change_sum++;
327         }
328         if(lcd_view == 1)
329         {
330             R_K_mode = !R_K_mode;//切换参数
331         }
332 
333     }
334 
335     if(pwm_change_flag == 1 && uwTick-pwm_mode_Tick>5000) pwm_mode = !pwm_mode;//5秒后，pwm模式切换
336     if(uwTick-pwm_mode_Tick>5000) pwm_change_flag = 0;//切换完成，过程标志拉低
337 
338 
339     if(key[2].short_flag == 1)
340     {
341         key[2].short_flag = 0;
342         if(lcd_view == 1)
343         {
344             if(R_K_mode == 0)
345             {
346                 if(R_value++ == 10) R_value = 1;//加
347             }
348             else
349             {
350                 if(K_value++ == 10) K_value = 1;
351             }
352 
353         }
354     }
```

```
355
356     if(key[3].short_flag == 1)
357     {
358         key[3].short_flag = 0;
359 
360         if(lcd_view == 1)//参数界面
361         {
362             if(R_K_mode == 0)
363             {
364                 if(R_value-- == 1) R_value = 10;//减
365             }
366             else
367             {
368                 if(K_value-- == 1) K_value = 10;
369             }
370         }
371         if(lcd_view == 0)
372         {
373             pa1_duty_lock = 0;//解锁
374         }
375     }
376 
377     if(lcd_view == 0)//数据界面
378     {
379         if(key[3].long_flag == 1)
380         {
381             key[3].long_flag = 0;
382             pa1_duty_lock = 1;//上锁
383 
384         }
385 
386     }
387 
388 
389 }
390 void lcd_proc()
391 {
392     if(lcd_view == 0)//数据界面
393     {
394     sprintf(lcd_arry,"        DATA");
395     LCD_DisplayStringLine(Line1, (unsigned char *)lcd_arry);
396     if(pwm_mode == 0)
397     {
398         sprintf(lcd_arry,"     M=L");
399         LCD_DisplayStringLine(Line3, (unsigned char *)lcd_arry);
400     }
401     else
402     {
403         sprintf(lcd_arry,"     M=H");
404         LCD_DisplayStringLine(Line3, (unsigned char *)lcd_arry);
405     }
406     sprintf(lcd_arry,"     P=%-2d%%",pa1_duty);
407     LCD_DisplayStringLine(Line4, (unsigned char *)lcd_arry);
408     sprintf(lcd_arry,"     V=%-6.1f",speed_value);
409     LCD_DisplayStringLine(Line5, (unsigned char *)lcd_arry);
410     }
411     if(lcd_view == 1)//参数界面
412     {
413     sprintf(lcd_arry,"        PARA");
414     LCD_DisplayStringLine(Line1, (unsigned char *)lcd_arry);
415     sprintf(lcd_arry,"     R=%-2d",R_value);
416     LCD_DisplayStringLine(Line3, (unsigned char *)lcd_arry);
417     sprintf(lcd_arry,"     K=%-2d",K_value);
418     LCD_DisplayStringLine(Line4, (unsigned char *)lcd_arry);
419     }
420     if(lcd_view == 2)//统计界面
421     {
422     sprintf(lcd_arry,"        RECD");
423     LCD_DisplayStringLine(Line1, (unsigned char *)lcd_arry);
424     sprintf(lcd_arry,"     N=%-2d",mode_change_sum);
425     LCD_DisplayStringLine(Line3, (unsigned char *)lcd_arry);
```

```
426       sprintf(lcd_arry,"     MH=%-6.1f",H_max_speed);
427       LCD_DisplayStringLine(Line4, (unsigned char *)lcd_arry);
428       sprintf(lcd_arry,"     ML=%-6.1f",L_max_speed);
429       LCD_DisplayStringLine(Line5, (unsigned char *)lcd_arry);
430       }
431
432
433   }
434
435   /* USER CODE END 4 */
436
437   /**
438     * @brief  This function is executed in case of error occurrence.
439     * @retval None
440     */
441   void Error_Handler(void)
442   {
443     /* USER CODE BEGIN Error Handler Debug */
444       /* User can add his own implementation to report the HAL error return state */
445
446     /* USER CODE END Error_Handler_Debug */
447   }
448
449   #ifdef  USE_FULL_ASSERT
450   /**
451     * @brief  Reports the name of the source file and the source line number
452     *         where the assert_param error has occurred.
453     * @param  file: pointer to the source file name
454     * @param  line: assert param error line source number
455     * @retval None
456     */
457   void assert_failed(uint8_t *file, uint32_t line)
458   {
459     /* USER CODE BEGIN 6 */
460       /* User can add his own implementation to report the file name and line number,
461          tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
462     /* USER CODE END 6 */
463   }
464   #endif /* USE_FULL_ASSERT */
465
```