

```

1  /* USER CODE BEGIN Header */
2  /**
3   *
4   * @file      : main.c
5   * @brief     : Main program body
6   *
7   * @attention
8   *
9   * Copyright (c) 2024 STMicroelectronics.
10  * All rights reserved.
11  *
12  * This software is licensed under terms that can be found in the LICENSE file
13  * in the root directory of this software component.
14  * If no LICENSE file comes with this software, it is provided AS-IS.
15  *
16  *
17  */
18  /* USER CODE END Header */
19  /* Includes -----*/
20  #include "main.h"
21  #include "adc.h"
22  #include "dma.h"
23  #include "tim.h"
24  #include "usart.h"
25  #include "gpio.h"
26
27  /* Private includes -----*/
28  /* USER CODE BEGIN Includes */
29  #include "led.h"
30  #include "interrupt.h"
31  #include "stdio.h"
32  #include "string.h"
33  #include "lcd.h"
34  #include "i2c_hal.h"
35  #include "seg.h"
36  #include "ds18b20.h"
37  #include "dht11.h"
38
39
40  /* USER CODE END Includes */
41
42  /* Private typedef -----*/
43  /* USER CODE BEGIN PTD */
44  extern struct keys key[4];
45  extern char rx_data;
46  extern char rx_array[50];
47  extern char rx_pointer;
48  extern uint PA1_freq, PA1_duty;
49
50  /* USER CODE END PTD */
51
52  /* Private define -----*/
53  /* USER CODE BEGIN PD */
54
55  /* USER CODE END PD */
56
57  /* Private macro -----*/
58  /* USER CODE BEGIN PM */
59
60  /* USER CODE END PM */
61
62  /* Private variables -----*/
63
64  /* USER CODE BEGIN PV */
65  char lcd_array[50];
66  char lcd_view;
67  uint16 ADC1_array[2];
68  uint16 ADC2_array[3];
69  uint PA7_freq = 1000;
70  uint PA7_duty = 50;
71  uint PA7_autoreload, PA7_compare;

```

```

72
73 char LCD_view2_REC_PA_flag;//0, PA4
74 int PA4_vol_array[150];
75 int PA5_vol_array[150];
76 IO uint32_t key_uwTick;
77 //char PARA_X_value;
78 //char PARA_Y_value;
79
80 int PARA_X_value;
81 int PARA_Y_value;
82
83 char vol_measure_flag;
84 char pwm_mode_flag;//0:倍频      1: 分频
85 char clear_record_flag;
86
87 IO uint32_t eeprom_uwTick;
88 int REC_PA4_N_value;
89 float REC_PA4_A_value, REC_PA4_T_value, REC_PA4_H_value;
90 int REC_PA5_N_value;
91 float REC_PA5_A_value, REC_PA5_T_value, REC_PA5_H_value;
92
93 char lcd_mode_flag;//0, 正向; 1, 翻转
94 IO uint32_t led_uwTick;
95 IO uint32_t pwm_uwTick;
96
97 char led_num;
98
99 char test_0;
100 char test_1;
101
102
103 /* USER CODE END PV */
104
105 /* Private function prototypes -----*/
106 void SystemClock_Config(void);
107 /* USER CODE BEGIN PFP */
108
109 /* USER CODE END PFP */
110
111 /* Private user code -----*/
112 /* USER CODE BEGIN 0 */
113 void key_proc(void);
114 void rx_proc(void);
115 void lcd_proc(void);
116 void eeprom_proc(void);
117 void pwm_proc(void);
118 void vol_measure(void);
119 void led_proc(void);
120 /* USER CODE END 0 */
121
122 /**
123  * @brief The application entry point.
124  * @retval int
125  */
126 int main(void)
127 {
128     /* USER CODE BEGIN 1 */
129
130     /* USER CODE END 1 */
131
132     /* MCU Configuration-----*/
133
134     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
135     HAL_Init();
136
137     /* USER CODE BEGIN Init */
138
139     /* USER CODE END Init */
140
141     /* Configure the system clock */
142     SystemClock_Config();

```

```

143
144 /* USER CODE BEGIN SysInit */
145
146 /* USER CODE END SysInit */
147
148 /* Initialize all configured peripherals */
149 MX_GPIO_Init();
150 MX_DMA_Init();
151 MX_TIM6_Init();
152 MX_USART1_UART_Init();
153 MX_TIM8_Init();
154 MX_TIM16_Init();
155 MX_ADC1_Init();
156 MX_ADC2_Init();
157 MX_TIM2_Init();
158 MX_TIM3_Init();
159 MX_TIM17_Init();
160 /* USER CODE BEGIN 2 */
161
162 /* USER CODE END 2 */
163
164 /* Infinite loop */
165 /* USER CODE BEGIN WHILE */
166 led_disp(0x00);
167
168 HAL_TIM_Base_Start_IT(&htim6);
169
170 HAL_UART_Receive_IT(&huart1, (uint8_t *)&rx_data, 1);
171
172 LCD_Init();
173 LCD_Clear(Black);
174 LCD_SetTextColor(White);
175 LCD_SetBackColor(Black);
176
177 I2CInit();
178 //
179 HAL_TIM_IC_Start_IT(&htim16, TIM_CHANNEL_1);
180 HAL_TIM_IC_Start_IT(&htim8, TIM_CHANNEL_1);
181 HAL_TIM_IC_Start_IT(&htim8, TIM_CHANNEL_2);
182 HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_2);
183
184 HAL_ADC_Start_DMA(&hadc1, (uint32_t *)ADC1_array, 2);
185
186 HAL_ADC_Start_DMA(&hadc2, (uint32_t *)ADC2_array, 3);
187
188 HAL_TIM_PWM_Start(&htim17, TIM_CHANNEL_1);
189
190 led_disp(0x00);
191
192 test_0 = eeprom_read(0);
193 test_1 = eeprom_read(1);
194
195 while (1)
196 {
197     /* USER CODE END WHILE */
198
199     /* USER CODE BEGIN 3 */
200     key_proc();
201     rx_proc();
202     lcd_proc();
203     pwm_proc();
204 //     eeprom_proc();
205     led_proc();
206     vol_measure();
207 }
208 /* USER CODE END 3 */
209 }
210
211 /**
212  * @brief System Clock Configuration
213  * @retval None

```

```

214  */
215 void SystemClock_Config(void)
216 {
217     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
218     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
219
220     /** Configure the main internal regulator output voltage
221     */
222     HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);
223
224     /** Initializes the RCC Oscillators according to the specified parameters
225     * in the RCC_OscInitTypeDef structure.
226     */
227     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
228     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
229     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
230     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
231     RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV3;
232     RCC_OscInitStruct.PLL.PLLN = 20;
233     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
234     RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
235     RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
236     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
237     {
238         Error_Handler();
239     }
240
241     /** Initializes the CPU, AHB and APB buses clocks
242     */
243     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
244                                 | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
245     RCC_ClkInitStruct.SYSClkSource = RCC_SYSCLKSOURCE_PLLCLK;
246     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
247     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
248     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
249
250     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
251     {
252         Error_Handler();
253     }
254 }
255
256 /* USER CODE BEGIN 4 */
257 void led_proc(void)
258 {
259     if(uwTick - led_uwTick < 100) return;
260     led_uwTick = uwTick;
261
262     if(pwm_mode_flag == 0) //倍频
263         led_num |= 0x01;
264     else led_num &= 0xfe;
265
266     if(pwm_mode_flag == 1) //分频
267         led_num |= 0x02;
268     else led_num &= 0xfd;
269
270     if(ADC2_array[1] > ADC2_array[2]*eeprom_read(1))
271         led_num = 0x04;
272     else led_num &= 0xfb;
273
274     if(led_mode_flag == 0) //正向
275         led_num |= 0x08;
276     else led_num &= 0xf7;
277
278     led_disp(led_num);
279 }
280
281 void key_proc(void)
282 {
283     if(uwTick - key_uwTick < 50) return;
284     key_uwTick = uwTick;

```

```

285     for(int i=0;i<4;i++)
286     {
287         if(key[i].short_flag == 1 || key[i].long_flag == 1)
288             LCD_Clear(Black);
289     }
290
291     if(key[0].short_flag == 1)
292     {
293         key[0].short_flag = 0;
294         lcd_view++;
295         if(lcd_view == 3) lcd_view = 0;
296         if(lcd_view == 2) LCD_view2_REC_PA_flag = 0;
297     }
298
299
300     if(key[1].short_flag == 1)
301     {
302         key[1].short_flag = 0;
303         PARA_X_value++;
304         if(PARA_X_value == 5) PARA_X_value = 1;
305
306         eeprom_write(1,PARA_X_value);
307     }
308
309     if(key[2].short_flag == 1)
310     {
311         key[2].short_flag = 0;
312         PARA_Y_value++;
313         if(PARA_Y_value == 5) PARA_Y_value = 1;
314
315         eeprom_write(0,PARA_Y_value);
316     }
317
318     if(key[3].short_flag == 1)
319     {
320         key[3].short_flag = 0;
321         if(lcd_view == 0)
322         {
323             vol_measure_flag = 1;
324         }
325
326         if(lcd_view == 1)
327         {
328             pwm_mode_flag = !pwm_mode_flag;
329         }
330
331         if(lcd_view == 2)
332         {
333             LCD_view2_REC_PA_flag = !LCD_view2_REC_PA_flag;
334         }
335     }
336
337     if(key[3].long_flag == 1)
338     {
339         key[3].long_flag = 0;
340         clear_record_flag = 1;
341     }
342
343
344 }
345
346 void pwm_proc(void)
347 {
348     if(uwTick - pwm_uwTick < 100) return;
349     pwm_uwTick = uwTick;
350     if(pwm_mode_flag == 0) //倍频
351     {
352         PA7_freq = PA1_freq*PARA_X_value;
353     }
354     else //分频
355         PA7_freq = PA1_freq/PARA_X_value;

```

```

356 PA7_autoreload = 1000000/PA7_freq;
357 PA7_compare = PA7_autoreload*PA7_duty/100;
358 __HAL_TIM_SetAutoreload(&htim17,PA7_autoreload);
359 __HAL_TIM_SetCompare(&htim17,TIM_CHANNEL_1,PA7_compare);
360
361 }
362 //void eeprom_proc(void)
363 //{
364 //  if(uwTick - eeprom_uwTick < 100) return;
365 //  eeprom_uwTick = uwTick;
366 //  eeprom_write(0, PARA_Y_value); //
367 //  eeprom_write(1, PARA_X_value); //
368 //}
369
370 void vol_measure(void)
371 {
372     int PA4_max_reg = 0;
373     int PA4_min_reg = 4095;
374     int PA4_avg_reg = 0;
375
376     int PA5_max_reg = 0;
377     int PA5_min_reg = 4095;
378     int PA5_avg_reg = 0;
379
380     int PA4_sum_reg, PA5_sum_reg;
381
382     if(vol_measure_flag == 1)
383     {
384         vol_measure_flag = 0;
385         if(LCD_view2_REC_PA_flag == 0)
386         {
387             PA4_vol_array[REC_PA4_N_value++] = ADC2_array[1];
388             for(int i=0; i<REC_PA4_N_value; i++)
389             {
390                 if(PA4_vol_array[i] > PA4_max_reg)
391                     PA4_max_reg = PA4_vol_array[i];
392                 if(PA4_vol_array[i] < PA4_min_reg)
393                     PA4_min_reg = PA4_vol_array[i];
394                 PA4_sum_reg += PA4_vol_array[i];
395             }
396
397             PA4_avg_reg = PA4_sum_reg/REC_PA4_N_value;
398
399             REC_PA4_A_value = PA4_max_reg*3.3/4096;
400             REC_PA4_T_value = PA4_min_reg*3.3/4096;
401             REC_PA4_H_value = PA4_avg_reg*3.3/4096;
402
403         }
404         else if(LCD_view2_REC_PA_flag == 1)
405         {
406             [                ]++ = [2];
407             for(int i=0; i<REC_PA5_N_value; i++)
408             {
409                 if(PA5_vol_array[i] > PA5_max_reg)
410                     PA5_max_reg = PA5_vol_array[i];
411                 if(PA5_vol_array[i] < PA5_min_reg)
412                     PA5_min_reg = PA5_vol_array[i];
413                 PA5_sum_reg += PA5_vol_array[i];
414             }
415
416             PA5_avg_reg = PA5_sum_reg/REC_PA5_N_value;
417
418             REC_PA5_A_value = PA5_max_reg*3.3/4096;
419             REC_PA5_T_value = PA5_min_reg*3.3/4096;
420             REC_PA5_H_value = PA5_avg_reg*3.3/4096;
421
422         }
423     }
424
425     if(clear_record_flag == 1)
426

```

```

427 {
428     clear_record_flag = 0;
429     if(LCD_view2_REC_PA_flag == 0)
430     {
431         REC_PA4_N_value = 0;
432         REC_PA4_A_value = 0;
433         REC_PA4_T_value = 0;
434         REC_PA4_H_value = 0;
435     }
436
437     if(LCD_view2_REC_PA_flag == 1)
438     {
439         REC_PA5_N_value = 0;
440         REC_PA5_A_value = 0;
441         REC_PA5_T_value = 0;
442         REC_PA5_H_value = 0;
443     }
444 }
445
446 void lcd_proc(void)
447 {
448
449     if(lcd_view == 0)
450     {
451         sprintf(lcd_array, "DATA");
452         LCD_DisplayStringLine(Line1, (u8 *)lcd_array);
453
454
455         sprintf(lcd_array, "PA4=%-4.2f", ADC2_array[1]*3.3/4096);
456         LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
457
458         sprintf(lcd_array, "PA5=%-4.2f", ADC2_array[2]*3.3/4096);
459         LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
460
461         sprintf(lcd_array, "PA1=%-6d", PA1_freq);
462         LCD_DisplayStringLine(Line5, (u8 *)lcd_array);
463     }
464     else if(lcd_view == 1)
465     {
466         sprintf(lcd_array, "PARAM");
467         LCD_DisplayStringLine(Line1, (u8 *)lcd_array);
468
469         sprintf(lcd_array, "X=%d", eeprom_read(1));
470         LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
471
472         sprintf(lcd_array, "Y=%d", eeprom_read(0));
473         LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
474     }
475     else if(lcd_view == 2)
476     {
477         if(LCD_view2_REC_PA_flag == 0)
478         {
479             sprintf(lcd_array, "REC-PA4");
480             LCD_DisplayStringLine(Line1, (u8 *)lcd_array);
481
482             sprintf(lcd_array, "N=%-4d", REC_PA4_N_value);
483             LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
484
485             sprintf(lcd_array, "A=%-4.2f", REC_PA4_A_value);
486             LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
487
488             sprintf(lcd_array, "T=%-4.2f", REC_PA4_T_value);
489             LCD_DisplayStringLine(Line5, (u8 *)lcd_array);
490
491             sprintf(lcd_array, "H=%-4.2f", REC_PA4_H_value);
492             LCD_DisplayStringLine(Line6, (u8 *)lcd_array);
493         }
494         else if(LCD_view2_REC_PA_flag == 1)
495         {
496             sprintf(lcd_array, "REC-PA5");
497             LCD_DisplayStringLine(Line1, (u8 *)lcd_array);

```

```

498     sprintf(lcd_array, "N=%-4d", REC_PA5_N_value);
499     LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
500
501     sprintf(lcd_array, "A=%-4.2f", REC_PA5_A_value);
502     LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
503
504     sprintf(lcd_array, "T=%-4.2f", REC_PA5_T_value);
505     LCD_DisplayStringLine(Line5, (u8 *)lcd_array);
506
507     sprintf(lcd_array, "H=%-4.2f", REC_PA5_H_value);
508     LCD_DisplayStringLine(Line6, (u8 *)lcd_array);
509 }
510
511 }
512
513
514
515
516 void rx_proc(void)
517 {
518     //判断数据是否接受完毕
519     if(rx_pointer != 0)
520     {
521         int temp = rx_pointer;
522         //接收一次数据需要9个Bit
523         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
524         //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
525         //显然这段时间小于1ms, 1ms能处理9bit, 间隔时间一定小于9bit,
526         if(temp == rx_pointer)
527         {
528             //串口接收处理部分
529             if(strcmp(rx_array, "X") == 0)
530                 printf("X:%d\n", eeprom_read(1));
531             else if(strcmp(rx_array, "Y") == 0)
532                 printf("Y:%d\n", eeprom_read(0));
533             else if(strcmp(rx_array, "PA1") == 0)
534                 printf("PA1:%d\n", PA1_freq);
535             else if(strcmp(rx_array, "PA4") == 0)
536                 printf("PA4:%.2f\n", ADC2_array[1]*3.3/4096);
537             else if(strcmp(rx_array, "PA5") == 0)
538                 printf("PA5:%.2f\n", ADC2_array[2]*3.3/4096);
539             else if(strcmp(rx_array, "#") == 0)
540                 lcd_mode_flag = !lcd_mode_flag;
541
542             if(lcd_mode_flag == 0)
543             {
544                 //上——>下, 左——>右
545                 LCD_WriteReg(R1, 0x0000); // set SS and SM bit //0x0100
546                 LCD_WriteReg(R96, 0xA700); // Gate Scan Line 0xA700
547                 LCD_Clear(Black);
548             }
549             else
550             {
551                 //下——>上, 右——>左
552                 LCD_WriteReg(R1, 0x0100); // set SS and SM bit //0x0100
553                 LCD_WriteReg(R96, 0xA700); // Gate Scan Line 0xA700
554                 LCD_Clear(Black);
555             }
556
557             printf("%s\n", rx_array);
558
559             rx_pointer=0;memset(rx_array, 0, 50);
560         }
561     }
562 }
563
564 int fputc(int ch, FILE *f)
565 {
566     /* Your implementation of fputc(). */
567     HAL_UART_Transmit(&huart1, (const uint8_t *)&ch, 1, 50);
568     return ch;
569 }

```



```
569
570
571 /* USER CODE END 4 */
572
573 /**
574  * @brief This function is executed in case of error occurrence.
575  * @retval None
576  */
577 void Error_Handler(void)
578 {
579     /* USER CODE BEGIN Error_Handler_Debug */
580     /* User can add his own implementation to report the HAL error return state */
581     __disable_irq();
582     while (1)
583     {
584     }
585     /* USER CODE END Error_Handler_Debug */
586 }
587
588 #ifdef USE_FULL_ASSERT
589 /**
590  * @brief Reports the name of the source file and the source line number
591  * where the assert_param error has occurred.
592  * @param file: pointer to the source file name
593  * @param line: assert_param error line source number
594  * @retval None
595  */
596 void assert_failed(uint8_t *file, uint32_t line)
597 {
598     /* USER CODE BEGIN 6 */
599     /* User can add his own implementation to report the file name and line number,
600      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
601     /* USER CODE END 6 */
602 }
603 #endif /* USE_FULL_ASSERT */
604
```