

```

1  /* USER CODE BEGIN Header */
2  /**
3   * *****
4   * @file      : main.c
5   * @brief     : Main program body
6   * *****
7   * @attention
8   *
9   * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
10  * All rights reserved.</center></h2>
11  *
12  * This software component is licensed by ST under BSD 3-Clause license,
13  * the "License"; You may not use this file except in compliance with the
14  * License. You may obtain a copy of the License at:
15  *          opensource.org/licenses/BSD-3-Clause
16  *
17  * *****
18  */
19  /* USER CODE END Header */
20  /* Includes -----*/
21  #include "main.h"
22  #include "adc.h"
23  #include "dma.h"
24  #include "tim.h"
25  #include "usart.h"
26  #include "gpio.h"
27
28  /* Private includes -----*/
29  /* USER CODE BEGIN Includes */
30  #include "lcd.h"
31  #include "stdio.h"
32  #include "string.h"
33  #include "interrupt.h"
34  #include "led.h"
35  #include "i2c_hal.h"
36
37  /* USER CODE END Includes */
38
39  /* Private typedef -----*/
40  /* USER CODE BEGIN PTD */
41  extern char rx_array[50];
42  extern char rx_data;
43  extern char rx_pointer;
44  extern uint PA15_freq, PA15_duty, PA15_rise, PA15_fall;
45  extern uint PB4_freq, PB4_duty, PB4_rise, PB4_fall;
46  extern struct keys key[4];
47
48  /* USER CODE END PTD */
49
50  /* Private define -----*/
51  /* USER CODE BEGIN PD */
52  #define PA1_FREQ 1000000
53
54  /* USER CODE END PD */
55
56  /* Private macro -----*/
57  /* USER CODE BEGIN PM */
58
59  /* USER CODE END PM */
60
61  /* Private variables -----*/
62
63  /* USER CODE BEGIN PV */
64  uint16_t adc1_array[1], adc2_array[1];
65  float adc2_vol;
66  uint PA1_autoreload, PA1_compare;
67  uint PA1_freq = 2000;
68  uint PA1_duty = 10;
69  char lcd_array[50];
70  char lcd_view;
71  char ctrl_mode = 0; //初始自动控制

```

```

72  char GEAR_num = 0;
73  char change_flag;
74  IO uint32_t key_uwTick;
75  IO uint32_t rx_uwTick;
76  IO uint32_t led_uwTick;
77  float temp_value;
78  char led_num;
79  char rx_right_flag;
80  char key_sleep_flag;
81  char rx_sleep_flag;
82  char sleep_flag;
83
84  /* USER CODE END PV */
85
86  /* Private function prototypes -----*/
87  void SystemClock_Config(void);
88  /* USER CODE BEGIN PFP */
89
90  /* USER CODE END PFP */
91
92  /* Private user code -----*/
93  /* USER CODE BEGIN 0 */
94  void rx_proc();
95  void pwm_proc();
96  void lcd_proc();
97  void key_proc();
98  void adc_proc();
99  void led_proc();
100 void sleep_proc();
101 /* USER CODE END 0 */
102
103 /**
104  * @brief The application entry point.
105  * @retval int
106  */
107 int main(void)
108 {
109     /* USER CODE BEGIN 1 */
110
111     /* USER CODE END 1 */
112
113     /* MCU Configuration-----*/
114
115     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
116     HAL_Init();
117
118     /* USER CODE BEGIN Init */
119
120     /* USER CODE END Init */
121
122     /* Configure the system clock */
123     SystemClock_Config();
124
125     /* USER CODE BEGIN SysInit */
126
127     /* USER CODE END SysInit */
128
129     /* Initialize all configured peripherals */
130     MX_GPIO_Init();
131     MX_DMA_Init();
132     MX_TIM2_Init();
133     MX_TIM3_Init();
134     MX_TIM6_Init();
135     MX_TIM15_Init();
136     MX_ADC2_Init();
137     MX_USART1_UART_Init();
138     MX_ADC1_Init();
139     MX_TIM8_Init();
140     /* USER CODE BEGIN 2 */
141
142     LCD_Init();

```

```

143 /* USER CODE END 2 */
144
145 /* Infinite loop */
146 /* USER CODE BEGIN WHILE */
147
148 LCD_Clear(Black);
149 LCD_SetBackColor(Black);
150 LCD_SetTextColor(White);
151
152 HAL_TIM_Base_Start_IT(&htim6); //定时器中断
153
154 HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2); //PWM
155
156 HAL_UART_Receive_IT(&huart1, (uint8_t *)&rx_data, 1); //串口接收中断
157
158 HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1); //捕获中断
159 HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_2);
160 HAL_TIM_IC_Start_IT(&htim8, TIM_CHANNEL_1);
161
162 HAL_ADC_Start_DMA(&hadc1, (uint32_t *)adc1_array, 1); //ADC_DMA
163 HAL_ADC_Start_DMA(&hadc2, (uint32_t *)adc2_array, 1);
164
165 while (1)
166 {
167 /* USER CODE END WHILE */
168
169 /* USER CODE BEGIN 3 */
170 rx_proc();
171 pwm_proc();
172 lcd_proc();
173 key_proc();
174 led_proc();
175 adc_proc();
176 sleep_proc();
177 }
178 /* USER CODE END 3 */
179 }
180
181 /**
182  * @brief System Clock Configuration
183  * @retval None
184  */
185 void SystemClock_Config(void)
186 {
187     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
188     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
189
190     /** Configure the main internal regulator output voltage
191     */
192     HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);
193
194     /** Initializes the RCC Oscillators according to the specified parameters
195     * in the RCC_OscInitTypeDef structure.
196     */
197     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
198     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
199     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
200     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
201     RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV3;
202     RCC_OscInitStruct.PLL.PLLN = 20;
203     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
204     RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
205     RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
206     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
207     {
208         Error_Handler();
209     }
210
211     /** Initializes the CPU, AHB and APB buses clocks
212     */
213     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLOCK

```

```

214 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
215 RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
216 RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
217 RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
218 RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
219
220 if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
221 {
222     Error_Handler();
223 }
224
225
226 /* USER CODE BEGIN 4 */
227 void sleep_proc()
228 {
229     if(uwTick-key_uwTick>5000)//5s内按键无动作
230         key_sleep_flag = 1;
231     else key_sleep_flag = 0;
232
233     if(uwTick-rx_uwTick>5000)//5s内串口未下发命令
234         rx_sleep_flag = 1;
235     else rx_sleep_flag = 0;
236
237     if(key_sleep_flag&&rx_sleep_flag)
238         lcd_view = 1;
239
240
241 }
242 void led_proc()
243 {
244     if(ctrl_mode == 0)
245         led_num |=0x80;//亮
246     else
247         led_num &=0x7f;//灭
248
249     if(GEAR_num == 1)
250         led_num |=0x01;//亮
251     else led_num &=0xfe;//灭
252
253     if(GEAR_num == 2)
254         led_num |=0x02;//亮
255     else led_num &=0xfd;//灭
256
257     if(GEAR_num == 3)
258         led_num |=0x04;//亮
259     else led_num &=0xfb;//灭
260
261     if(rx_right_flag == 1)
262     {
263         led_num |=0x08;//亮
264         if(uwTick-led_uwTick>3000)
265         {
266             rx_right_flag = 0;
267             led_num &=0xf7;//灭
268         }
269     }
270
271     led_disp(led_num);
272
273
274 }
275 void adc_proc()
276 {
277     adc2_vol = adc2_array[0]*3.3/4096;
278     if(ctrl_mode == 0)//自动控制模式
279     {
280         if(adc2_vol<1.5)
281             GEAR_num = 1;
282         else if(adc2_vol<2)
283             GEAR_num = 2;
284         else GEAR_num = 3;

```

```

285     }
286
287     temp_value = 10*adc2_vol + 10;
288     if(temp_value>40)
289         temp_value = 40;
290
291 }
292 void key_proc ()
293 {
294     for(int i=0;i<4;i++)
295         if(key[i].short_flag == 1)
296         {
297             LCD_Clear(Black);
298             key_uwTick = uwTick;
299         }
300
301     if(key[0].short_flag == 1)
302     {
303         key[0].short_flag = 0;
304         if(lcd_view == 0)
305         {
306             ctrl_mode = !ctrl_mode;
307         }
308         else change_flag = 1;
309     }
310
311     if(key[1].short_flag == 1)
312     {
313         key[1].short_flag = 0;
314         if(lcd_view == 0)//数据界面
315         {
316             if(ctrl_mode == 1)//手动控制
317             {
318                 GEAR_num++;
319                 if(GEAR_num>3)
320                     GEAR_num = 3;
321             }
322         }
323         else
324             change_flag = 1;
325     }
326
327     if(key[2].short_flag == 1)
328     {
329         key[2].short_flag = 0;
330         if(lcd_view == 0)//数据界面
331         {
332             if(ctrl_mode == 1)//手动控制
333             {
334                 GEAR_num--;
335                 if(GEAR_num<1)
336                     GEAR_num = 1;
337             }
338         }
339         else
340             change_flag = 1;
341     }
342
343     if(key[3].short_flag == 1)
344         key[3].short_flag = 0;
345
346     if(change_flag == 1)
347     {
348         lcd_view = 0;//切换到数据界面
349         change_flag = 0;//标志位清零
350     }
351 }
352
353 void lcd_proc ()
354 {
355     if(lcd_view == 0)//

```

```

356 {
357     sprintf(lcd_array, "          DATA");
358     LCD_DisplayStringLine(Line1, (u8 *)lcd_array);
359     sprintf(lcd_array, "          TEMP:%-4.1f", temp_value);
360     LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
361     if(ctrl_mode == 0)
362         sprintf(lcd_array, "          MODE:Auto");
363     else sprintf(lcd_array, "          MODE:Manu");
364     LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
365     sprintf(lcd_array, "          GEAR:%d", GEAR_num);
366     LCD_DisplayStringLine(Line5, (u8 *)lcd_array);
367 }
368
369 else
370 {
371     sprintf(lcd_array, "          ");
372     LCD_DisplayStringLine(Line1, (u8 *)lcd_array);
373     sprintf(lcd_array, "          ");
374     LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
375     sprintf(lcd_array, "          SLEEPING ");
376     LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
377     sprintf(lcd_array, "          TEMP:%-4.1f", temp_value);
378     LCD_DisplayStringLine(Line5, (u8 *)lcd_array);
379 }
380
381 }
382 void pwm_proc ()
383 {
384     if(GEAR_num == 1)
385         PA1_duty = 10;
386
387     if(GEAR_num == 2)
388         PA1_duty = 40;
389
390     if(GEAR_num == 3)
391         PA1_duty = 80;
392
393     PA1_autoreload = PA1_FREQ/PA1_freq;
394     PA1_compare = PA1_autoreload*PA1_duty/100;
395
396     __HAL_TIM_SET_AUTORELOAD(&htim2, PA1_autoreload);
397     __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, PA1_compare);
398
399 }
400 void rx_proc ()
401 {
402     if(strcmp(rx_array, "B1") == 0 || strcmp(rx_array, "B2") == 0 || strcmp(rx_array, "B3") == 0)
403     {
404         = 1;
405         led_uwTick = uwTick;
406         rx_uwTick = uwTick;
407     }
408
409     if(rx_pointer!=0)
410     {
411         int temp = rx_pointer;
412         HAL_Delay(1);
413         if(temp==rx_pointer)
414         {
415             printf("array is :%s", rx_array);
416             if(rx_pointer == 2)
417             {
418                 if(strcmp(rx_array, "B1") == 0)
419                 {
420                     if(lcd_view == 0)
421                         key[0].short_flag = 1;
422                     else change_flag = 1;
423                 }
424
425                 else if(strcmp(rx_array, "B2") == 0)
426                     if(lcd_view == 0)

```

```

427         key[1].short_flag = 1;
428         else change_flag = 1;
429         else if(strcmp(rx_array,"B3") == 0)
430             if(lcd_view == 0)
431                 key[2].short_flag = 1;
432                 else change_flag = 1;
433                 else printf("NULL");
434
435     }
436     else printf("NULL");
437     rx_pointer = 0;memset(rx_array,0,50);
438 }
439 }
440
441
442
443 int fputc(int ch, FILE *f)
444 {
445     HAL_UART_Transmit(&huart1, (const uint8_t *)&ch, 1,20);
446     return ch;
447 }
448
449
450 /* USER CODE END 4 */
451
452 /**
453  * @brief This function is executed in case of error occurrence.
454  * @retval None
455  */
456 void Error_Handler(void)
457 {
458     /* USER CODE BEGIN Error_Handler_Debug */
459     /* User can add his own implementation to report the HAL error return state */
460
461     /* USER CODE END Error_Handler_Debug */
462 }
463
464 #ifdef USE_FULL_ASSERT
465 /**
466  * @brief Reports the name of the source file and the source line number
467  * where the assert_param error has occurred.
468  * @param file: pointer to the source file name
469  * @param line: assert_param error line source number
470  * @retval None
471  */
472 void assert_failed(uint8_t *file, uint32_t line)
473 {
474     /* USER CODE BEGIN 6 */
475     /* User can add his own implementation to report the file name and line number,
476        tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
477     /* USER CODE END 6 */
478 }
479 #endif /* USE_FULL_ASSERT */
480

```