

```

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  *
16  *         opensource.org/licenses/BSD-3-Clause
17  *
18  */
19  /* USER CODE END Header */
20  /* Includes -----*/
21  #include "main.h"
22  #include "tim.h"
23  #include "usart.h"
24  #include "gpio.h"
25
26  /* Private includes -----*/
27  /* USER CODE BEGIN Includes */
28  #include "stdio.h"
29  #include "led.h"
30  #include "interrupt.h"
31  #include "string.h"
32
33  /* USER CODE END Includes */
34
35  /* Private typedef -----*/
36  /* USER CODE BEGIN PTD */
37
38  /* USER CODE END PTD */
39
40  /* Private define -----*/
41  /* USER CODE BEGIN PD */
42  #define PA1_FREQ 1000000
43  #define PA7_FREQ 1000000
44  /* USER CODE END PD */
45
46  /* Private macro -----*/
47  /* USER CODE BEGIN PM */
48  extern uchar rx_array[50];
49  extern uchar rx_data;
50  extern uchar rx_pointer;
51  extern struct keys key[4];
52
53
54  /* USER CODE END PM */
55
56  /* Private variables -----*/
57
58  /* USER CODE BEGIN PV */
59  char lcd_array[50];
60  uchar lcd_view;
61  uint pal_freq = 1000;
62  uint pal_duty = 10;
63  uint pa7_freq = 1000;
64  uint pa7_duty = 10;
65  uchar ctrl_mode = 0;
66  char led_num;
67  IO uint32_t led_uwTick;
68  IO uint32_t pwm_uwTick;
69  uint pal_autoreload, pal_compare;
70  uint pa7_autoreload, pa7_compare;
71

```

```

72  /* USER CODE END PV */
73
74  /* Private function prototypes -----*/
75  void SystemClock_Config(void);
76  /* USER CODE BEGIN PFP */
77
78  /* USER CODE END PFP */
79
80  /* Private user code -----*/
81  /* USER CODE BEGIN 0 */
82  void rx_proc(void);
83  void lcd_proc(void);
84  void key_proc(void);
85  void led_proc(void);
86  void pwm_proc(void);
87
88  /* USER CODE END 0 */
89
90  /**
91   * @brief The application entry point.
92   * @retval int
93   */
94  int main(void)
95  {
96      /* USER CODE BEGIN 1 */
97
98      /* USER CODE END 1 */
99
100     /* MCU Configuration-----*/
101
102     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
103     HAL_Init();
104
105     /* USER CODE BEGIN Init */
106
107     /* USER CODE END Init */
108
109     /* Configure the system clock */
110     SystemClock_Config();
111
112     /* USER CODE BEGIN SysInit */
113
114     /* USER CODE END SysInit */
115
116     /* Initialize all configured peripherals */
117     MX_GPIO_Init();
118     MX_TIM2_Init();
119     MX_TIM6_Init();
120     MX_TIM17_Init();
121     MX_USART1_UART_Init();
122     /* USER CODE BEGIN 2 */
123
124     LCD_Init();
125     /* USER CODE END 2 */
126
127     /* Infinite loop */
128     /* USER CODE BEGIN WHILE */
129
130     LCD_Clear(Black);
131     LCD_SetBackColor(Black);
132     LCD_SetTextColor(White);
133
134     HAL_TIM_Base_Start_IT(&htim6); //开启按键中断
135     HAL_TIM_PWM_Start(&htim17, TIM_CHANNEL_1); //开启PWM输出
136     HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2); //开启PWM输出
137
138     HAL_UART_Receive_IT(&huart1, (uint8_t *)&rx_data, 1); //开启串口接收中断
139
140     while (1)
141     {
142         /* USER CODE END WHILE */

```

```

143
144     /* USER CODE BEGIN 3 */
145
146     if(rx_pointer!=0)
147     {
148         char temp = rx_pointer;
149         HAL_Delay(1);
150         if(temp == rx_pointer)
151             rx_proc();
152     }
153
154     key_proc();
155     lcd_proc();
156     led_proc();
157     pwm_proc();
158
159 }
160 /* USER CODE END 3 */
161
162
163 /**
164  * @brief System Clock Configuration
165  * @retval None
166  */
167 void SystemClock_Config(void)
168 {
169     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
170     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
171
172     /** Configure the main internal regulator output voltage
173     */
174     HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);
175
176     /** Initializes the RCC Oscillators according to the specified parameters
177     * in the RCC_OscInitTypeDef structure.
178     */
179     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
180     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
181     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
182     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
183     RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV3;
184     RCC_OscInitStruct.PLL.PLLN = 20;
185     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
186     RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
187     RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
188     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
189     {
190         Error_Handler();
191     }
192
193     /** Initializes the CPU, AHB and APB buses clocks
194     */
195     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
196                                     | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
197     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
198     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
199     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
200     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
201
202     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) != HAL_OK)
203     {
204         Error_Handler();
205     }
206 }
207
208 /* USER CODE BEGIN 4 */
209 void pwm_proc()
210 {
211     if(uwTick - pwm_uwTick < 100)
212         return;
213     pwm_uwTick = uwTick;

```

```

214     pal_autoreload = PA1_FREQ/pal_freq;
215     pal_compare = pal_duty*pal_autoreload/100;
216     pa7_autoreload = PA7_FREQ/pa7_freq;
217     pa7_compare = pa7_duty*pa7_autoreload/100;
218
219     __HAL_TIM_SetAutoreload(&htim2, pal_autoreload);
220     __HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_2, pal_compare);
221
222     __HAL_TIM_SetAutoreload(&htim17, pa7_autoreload);
223     __HAL_TIM_SetCompare(&htim17, TIM_CHANNEL_1, pa7_compare);
224
225 }
226
227 void led_proc()
228 {
229     if(uwTick-led_uwTick<100)
230         return;
231     led_uwTick = uwTick;
232
233
234     if(ctrl_mode == 0)
235         led_num = led_num|0x04;//LED3亮
236     else led_num = led_num&0xfb;//LED3灭
237
238     if(pal_freq>pa7_freq) //优先判断
239         led_num = led_num^0x01;//LED1翻转闪烁
240     else if(pal_freq<pa7_freq) //条件级高
241         led_num = led_num^0x02;//LED2翻转闪烁
242     else if(lcd_view == 0)
243     {
244         led_num = led_num|0x01;//LED1亮
245         led_num = led_num&0xfd;//LED2灭
246     }
247
248     else if(lcd_view == 1)
249     {
250         led_num = led_num|0x02;//LED2亮
251         led_num = led_num&0xfe;//LED1灭
252     }
253
254
255
256
257     led_disp(led_num);
258
259 }
260
261 void key_proc()
262 {
263     for(int i=0;i<4;i++)
264     {
265         if(key[i].short_flag == 1)
266             LCD_Clear(Black);
267     }
268     if(key[0].short_flag == 1)
269     {
270         key[0].short_flag = 0;
271
272         if(lcd_view == 0)
273         {
274             pal_freq = pal_freq + 1000;
275             if(pal_freq > 10000)
276                 pal_freq = 1000;
277         }
278         else
279         {
280             pa7_freq = pa7_freq + 1000;
281             if(pa7_freq > 10000)
282                 pa7_freq = 1000;
283         }
284     }

```

```

285     if(key[1].short_flag == 1)
286     {
287         key[1].short_flag = 0;
288
289         if(lcd_view == 0)
290         {
291             pal_duty = pal_duty + 10;
292             if(pal_duty > 90)
293                 pal_duty = 10;
294         }
295         else
296         {
297             pa7_duty = pa7_duty + 10;
298             if(pa7_duty > 90)
299                 pa7_duty = 10;
300         }
301     }
302 }
303
304 if(key[2].short_flag == 1)
305 {
306     key[2].short_flag = 0;
307     if(ctrl_mode == 0)
308         lcd_view = !lcd_view;
309 }
310
311 if(key[3].short_flag == 1)
312 {
313     key[3].short_flag = 0;
314     ctrl_mode = !ctrl_mode;
315 }
316 }
317
318 void lcd_proc()
319 {
320     if(lcd_view == 0)//PA1数据界面
321     {
322         sprintf(lcd_array, "    PA1");
323         LCD_DisplayStringLine(Line2, (u8 *)lcd_array);
324         sprintf(lcd_array, "    F:%-dHZ", pal_freq);
325         LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
326         sprintf(lcd_array, "    D:%-2d%", pal_duty);
327         LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
328     }
329     else
330     {
331         sprintf(lcd_array, "    PA7");
332         LCD_DisplayStringLine(Line2, (u8 *)lcd_array);
333         sprintf(lcd_array, "    F:%-dHZ", pa7_freq);
334         LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
335         sprintf(lcd_array, "    D:%-2d%", pa7_duty);
336         LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
337     }
338 }
339
340 void rx_proc()
341 {
342     if(ctrl_mode == 1)
343     {
344         if(rx_pointer == 1)
345         {
346             if(rx_array[0] == '@')
347                 lcd_view = 0;
348             else if(rx_array[0] == '#')
349                 lcd_view = 1;
350             else printf("ERROR\n");
351         }
352         else printf("ERROR\n");
353     }
354     else printf("KEY_CONTROL\n");
355 }

```

```

356     rx_pointer = 0;memset(rx_array,0,50);
357
358     int fputc(int ch, FILE *f)
359     {
360         HAL_UART_Transmit(&huart1, (const uint8_t *)&ch, 1, 20);
361         return ch;
362     }
363     /* USER CODE END 4 */
364
365     /**
366     * @brief This function is executed in case of error occurrence.
367     * @retval None
368     */
369     void Error_Handler(void)
370     {
371         /* USER CODE BEGIN Error_Handler_Debug */
372         /* User can add his own implementation to report the HAL error return state */
373
374         /* USER CODE END Error_Handler_Debug */
375     }
376
377     #ifdef USE_FULL_ASSERT
378     /**
379     * @brief Reports the name of the source file and the source line number
380     *        where the assert_param error has occurred.
381     * @param file: pointer to the source file name
382     * @param line: assert_param error line source number
383     * @retval None
384     */
385     void assert_failed(uint8_t *file, uint32_t line)
386     {
387         /* USER CODE BEGIN 6 */
388         /* User can add his own implementation to report the file name and line number,
389            tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
390         /* USER CODE END 6 */
391     }
392     #endif /* USE_FULL_ASSERT */
393

```