

```

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 "led.h"
32  #include "interrupt.h"
33  #include "stdio.h"
34  #include "string.h"
35  /* USER CODE END Includes */
36
37  /* Private typedef -----*/
38  /* USER CODE BEGIN PTD */
39
40  /* USER CODE END PTD */
41
42  /* Private define -----*/
43  /* USER CODE BEGIN PD */
44  /* USER CODE END PD */
45
46  /* Private macro -----*/
47  /* USER CODE BEGIN PM */
48  extern char rx_array[50];
49  extern char rx_data;
50  extern char rx_pointer;
51  extern struct keys key[4];
52  /* USER CODE END PM */
53
54  /* Private variables -----*/
55
56  /* USER CODE BEGIN PV */
57  char led_num;
58  uint16_t R37_array[1];
59  uint16_t R38_array[1];
60  float R37_vol, R38_vol;
61  char lcd_array[50];
62  char lcd_view;
63  char R37_check_flag, R38_check_flag;
64  float SR37_min = 1.2;
65  float SR37_max = 2.2;
66  float SR38_min = 1.4;
67  float SR38_max = 3.0;
68  uchar standard_state;
69  uchar change_state;
70  uchar clear_flag;
71  float R37_ok_rate;

```

```

72 uint R37_check_num,R37_ok_num;
73 float R38_ok_rate;
74 uint R38_check_num,R38_ok_num;
75 uint R37_ok_flag,R38_ok_flag;
76 uchar SR37_min_clear_flag;
77 uchar SR37_max_clear_flag;
78 uchar SR38_min_clear_flag;
79 uchar SR38_max_clear_flag;
80 IO uint32 t_led_uwTick;;
81 /* USER CODE END PV */
82
83 /* Private function prototypes -----*/
84 void SystemClock_Config(void);
85 /* USER CODE BEGIN PFP */
86
87 /* USER CODE END PFP */
88
89 /* Private user code -----*/
90 /* USER CODE BEGIN 0 */
91 void rx_proc();
92 void key_proc();
93 void led_proc();
94 void lcd_proc();
95 void rate_proc();
96 /* USER CODE END 0 */
97
98 /**
99  * @brief The application entry point.
100  * @retval int
101  */
102 int main(void)
103 {
104     /* USER CODE BEGIN 1 */
105
106     /* USER CODE END 1 */
107
108     /* MCU Configuration-----*/
109
110     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
111     HAL_Init();
112
113     /* USER CODE BEGIN Init */
114
115     /* USER CODE END Init */
116
117     /* Configure the system clock */
118     SystemClock_Config();
119
120     /* USER CODE BEGIN SysInit */
121
122     /* USER CODE END SysInit */
123
124     /* Initialize all configured peripherals */
125     MX_GPIO_Init();
126     MX_DMA_Init();
127     MX_ADC1_Init();
128     MX_ADC2_Init();
129     MX_TIM6_Init();
130     MX_USART1_UART_Init();
131     /* USER CODE BEGIN 2 */
132
133     LCD_Init();
134     /* USER CODE END 2 */
135
136     /* Infinite loop */
137     /* USER CODE BEGIN WHILE */
138
139     LCD_Clear(Black);
140     LCD_SetBackColor(Black);
141     LCD_SetTextColor(White);
142

```

```

143 HAL_ADC_Start_DMA(&hadc1, (uint32_t *)R38_array, 1);
144 HAL_ADC_Start_DMA(&hadc2, (uint32_t *)R37_array, 1);
145
146 HAL_TIM_Base_Start_IT(&htim6);
147
148 HAL_UART_Receive_IT(&huart1, (uint8_t *)&rx_data, 1);
149
150 while (1)
151 {
152     /* USER CODE END WHILE */
153
154     /* USER CODE BEGIN 3 */
155
156     if(rx_pointer!=0)
157     {
158         int temp = rx_pointer;
159         HAL_Delay(1);
160         if(temp == rx_pointer)
161             rx_proc();
162     }
163
164     key_proc();
165     led_proc();
166     lcd_proc();
167     rate_proc();
168 }
169 /* USER CODE END 3 */
170 }
171
172 /**
173  * @brief System Clock Configuration
174  * @retval None
175  */
176 void SystemClock_Config(void)
177 {
178     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
179     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
180
181     /** Configure the main internal regulator output voltage
182     */
183     HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);
184
185     /** Initializes the RCC Oscillators according to the specified parameters
186     * in the RCC_OscInitTypeDef structure.
187     */
188     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
189     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
190     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
191     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
192     RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV3;
193     RCC_OscInitStruct.PLL.PLLN = 20;
194     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
195     RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
196     RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
197     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
198     {
199         Error_Handler();
200     }
201
202     /** Initializes the CPU, AHB and APB buses clocks
203     */
204     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
205                                 | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
206     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
207     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
208     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
209     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
210
211     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
212     {
213

```

```

214     Error_Handler();
215 }
216 }
217
218 /* USER CODE BEGIN 4 */
219 void rate_proc()
220 {
221     R37_vol = R37_array[0]*3.3/4096;
222     R38_vol = R38_array[0]*3.3/4096;
223
224     if(R37_check_flag == 1)
225     {
226         R37_check_flag = 0;
227         if(R37_vol>=SR37_min&&R37_vol<=SR37_max)
228         {
229             R37_ok_num++;
230             R37_ok_flag = 1;
231             led_uwTick = uwTick;
232         }
233
234         R37_check_num++;
235         R37_ok_rate = R37_ok_num*100/R37_check_num;
236     }
237
238     if(R38_check_flag == 1)
239     {
240         R38_check_flag = 0;
241         if(R38_vol>=SR38_min&&R38_vol<=SR38_max)
242         {
243             R38_ok_num++;
244             R38_ok_flag = 1;
245             led_uwTick = uwTick;
246         }
247         R38_check_num++;
248         R38_ok_rate = R38_ok_num*100/R38_check_num;
249     }
250
251     if(clear_flag == 1)
252     {
253         clear_flag = 0;
254         R37_ok_rate = 0;
255         R38_ok_rate = 0;
256     }
257
258     if(SR37_max_clear_flag || SR37_min_clear_flag)
259     {
260         SR37_max_clear_flag = 0;
261         SR37_min_clear_flag = 0;
262         R37_ok_rate = 0;
263     }
264
265     if(SR38_max_clear_flag || SR38_min_clear_flag)
266     {
267         SR38_max_clear_flag = 0;
268         SR38_min_clear_flag = 0;
269         R38_ok_rate = 0;
270     }
271 }
272 void lcd_proc()
273 {
274     if(lcd_view == 0)
275     {
276         sprintf(lcd_array, "      GOODS");
277         LCD_DisplayStringLine(Line1, (u8 *)lcd_array);
278         sprintf(lcd_array, "      R37:%-4.2fV", R37_vol);
279         LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
280         sprintf(lcd_array, "      R38:%-4.2fV", R38_vol);
281         LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
282     }
283
284     if(lcd_view == 1)

```

```

285 {
286     sprintf(lcd_array, "        STANDARD");
287     LCD_DisplayStringLine(Line1, (u8 *)lcd_array);
288     sprintf(lcd_array, "        SR37:%-3.1f-%-3.1f", SR37_min, SR37_max);
289     LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
290     sprintf(lcd_array, "        SR38:%-3.1f-%-3.1f", SR38_min, SR38_max);
291     LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
292 }
293
294 if(lcd_view == 2)
295 {
296     sprintf(lcd_array, "        PASS");
297     LCD_DisplayStringLine(Line1, (u8 *)lcd_array);
298     sprintf(lcd_array, "        PR37:%-4.1f%%", R37_ok_rate);
299     LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
300     sprintf(lcd_array, "        PR38:%-4.1f%%", R38_ok_rate);
301     LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
302 }
303
304 void led_proc()
305 {
306     if(R37_ok_flag == 1)
307     {
308         led_num = led_num|0x01;
309         if(uwTick-led_uwTick>1000)
310         {
311             R37_ok_flag = 0;
312             led_num = led_num&0x0fe;
313         }
314     }
315     else led_num = led_num&0x0fe;
316
317     if(R38_ok_flag == 1)
318     {
319         led_num = led_num|0x02;
320
321         if(uwTick-led_uwTick>1000)
322         {
323             R38_ok_flag = 0;
324             led_num = led_num&0x0fd;
325         }
326     }
327
328     if(lcd_view == 0)
329         led_num = led_num|0x04;//3亮
330     else led_num = led_num&0xfb;//3灭
331
332     if(lcd_view == 1)
333         led_num = led_num|0x08;//4亮
334     else led_num = led_num&0xf7;//4灭
335
336     if(lcd_view == 2)
337         led_num = led_num|0x10;//5亮
338     else led_num = led_num&0xef;//5灭
339
340     led_disp(led_num);
341 }
342
343 void key_proc()
344 {
345     for(int i=0;i<4;i++)
346         if(key[i].short_flag==1)
347             LCD_Clear(Black);
348     if(key[0].short_flag == 1)
349     {
350         key[0].short_flag = 0;
351         lcd_view++;
352         if(lcd_view>2)
353             lcd_view = 0;
354     }
355     if(key[1].short_flag == 1)

```

```

356     key[1].short_flag = 0;
357     if(lcd_view == 0)//产品参数界面
358     {
359         R37_check_flag = 1;//R37合格率检测
360     }
361     if(lcd_view == 1)//标准设置界面
362     {
363         standard_state++;
364         if(standard_state>3)
365             standard_state = 0;
366     }
367 }
368
369 if(key[2].short_flag == 1)
370 {
371     key[2].short_flag = 0;
372     if(lcd_view == 0)
373     {
374         R38_check_flag = 1;//R38合格率检测
375     }
376     if(lcd_view == 1)
377     {
378         switch(standard_state)
379         {
380             case 0:
381                 SR37_max+=0.2;
382                 SR37_max_clear_flag = 1;
383                 if(SR37_max>3.0) SR37_max = 2.2;
384                 break;
385             case 1:
386                 SR37_min+=0.2;
387                 SR37_min_clear_flag = 1;
388                 if(SR37_min>2.0) SR37_min = 1.2;
389                 break;
390             case 2:
391                 SR38_max+=0.2;
392                 SR38_max_clear_flag = 1;
393                 if(SR38_max>3.0) SR38_max = 2.2;
394                 break;
395             case 3:
396                 SR38_min+=0.2;
397                 SR38_min_clear_flag = 1;
398                 if(SR38_min>2.0) SR38_min = 1.2;
399                 break;
400         }
401     }
402 }
403
404
405
406 if(key[3].short_flag == 1)
407 {
408     key[3].short_flag = 0;
409
410     if(lcd_view == 0)
411     {
412         clear_flag = 1;//清零合格率
413     }
414
415     if(lcd view == 1)
416     {
417         switch(standard_state)
418         {
419             case 0:
420                 SR37_max-=0.2;
421                 SR37_max_clear_flag = 1;
422                 if(SR37_max<2.2) SR37_max = 3.0;
423                 break;
424             case 1:
425                 SR37_min-=0.2;
426                 SR37_min_clear_flag = 1;

```

```

427         if(SR37_min<1.2) SR37_min = 2.0;
428         break;
429     case 2:
430         SR38_max-=0.2;
431         SR38_max_clear_flag = 1;
432         if(SR38_max<2.2) SR38_max = 3.0;
433         break;
434     case 3:
435         SR38_min-=0.2;
436         SR38_min_clear_flag = 1;
437         if(SR38_min<1.2) SR38_min = 2.0;
438         break;
439     }
440 }
441 }
442
443
444 void rx_proc()
445 {
446     if(rx_pointer == 3)
447     {
448         if(strcmp(rx_array, "R37") == 0)
449             printf("R37:%d,%d,%.1f%%", R37_check_num, R37_ok_num, R37_ok_rate);
450         if(strcmp(rx_array, "R38") == 0)
451             printf("R38:%d,%d,%.1f%%", R38_check_num, R38_ok_num, R38_ok_rate);
452     }
453     rx_pointer = 0;memset(rx_array, 0, 50);
454 }
455 int fputc(int ch, FILE *f)
456 {
457     HAL_UART_Transmit(&huart1, (const uint8_t *)&ch, 1, 20);
458     return ch;
459 }
460 /* USER CODE END 4 */
461
462 /**
463  * @brief This function is executed in case of error occurrence.
464  * @retval None
465  */
466 void Error_Handler(void)
467 {
468     /* USER CODE BEGIN Error_Handler_Debug */
469     /* User can add his own implementation to report the HAL error return state */
470
471     /* USER CODE END Error_Handler_Debug */
472 }
473
474 #ifdef USE_FULL_ASSERT
475 /**
476  * @brief Reports the name of the source file and the source line number
477  * where the assert_param error has occurred.
478  * @param file: pointer to the source file name
479  * @param line: assert_param error line source number
480  * @retval None
481  */
482 void assert_failed(uint8_t *file, uint32_t line)
483 {
484     /* USER CODE BEGIN 6 */
485     /* User can add his own implementation to report the file name and line number,
486        tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
487     /* USER CODE END 6 */
488 }
489 #endif /* USE_FULL_ASSERT */
490

```