****

A级达标线上测试报告



**学院** **专业**

**行政班级**

**学号**

**姓名**

**邮箱**

**手机** **完成日期**

**成绩**

# 题目名称：基于stm32f103微控制器的电炉温度控制仿真系统

一、题目要求

使用 stm32f103 微控制器，搭建一个闭环电炉控制系统。

二、设计思路

软件环境：Proteus8.8仿真软件、Keil软件开发系统、STM32CubeMX

系统框图：

显示

控制驱动

STM32F103

电炉

（oven）

AD

（cpu自带）

温度传感器

（电炉自带）

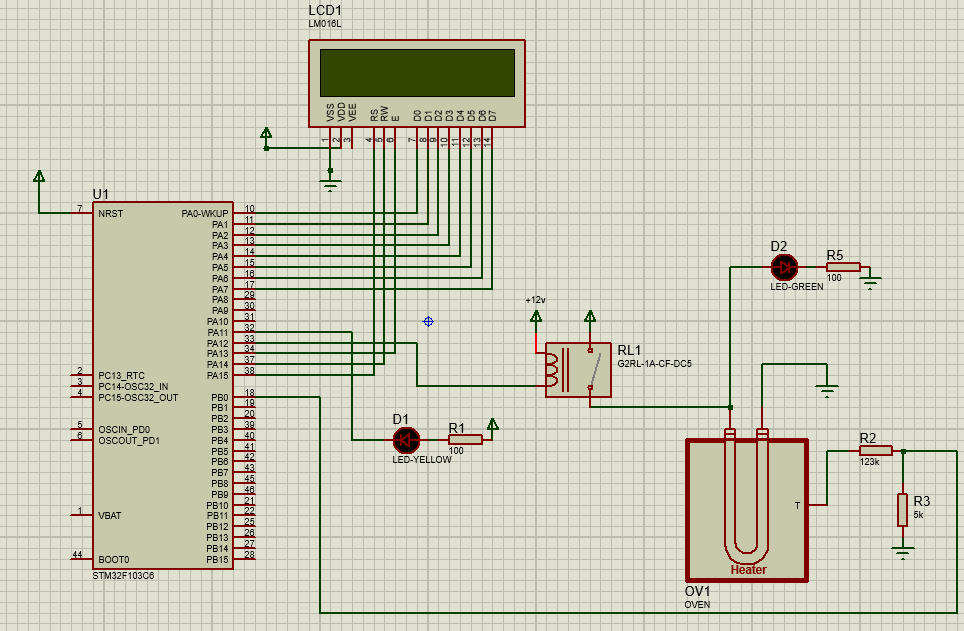
设计思路：

微控制器与LCD连接方式：

PA0---D0  
PA1---D1  
PA2---D2  
PA3---D3  
PA4---D4  
PA5---D5  
PA6---D6  
PA7---D7  
PA13---E  
PA14---RW  
PA15---RS

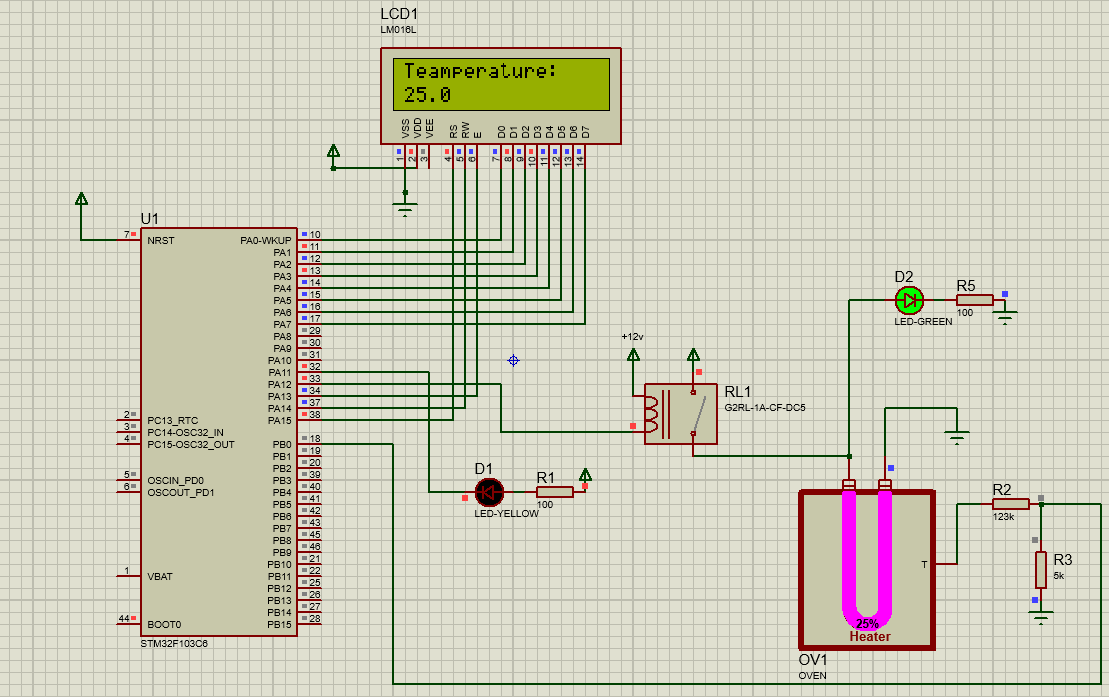
刷新时间指示灯与微控制器的PA11相连。

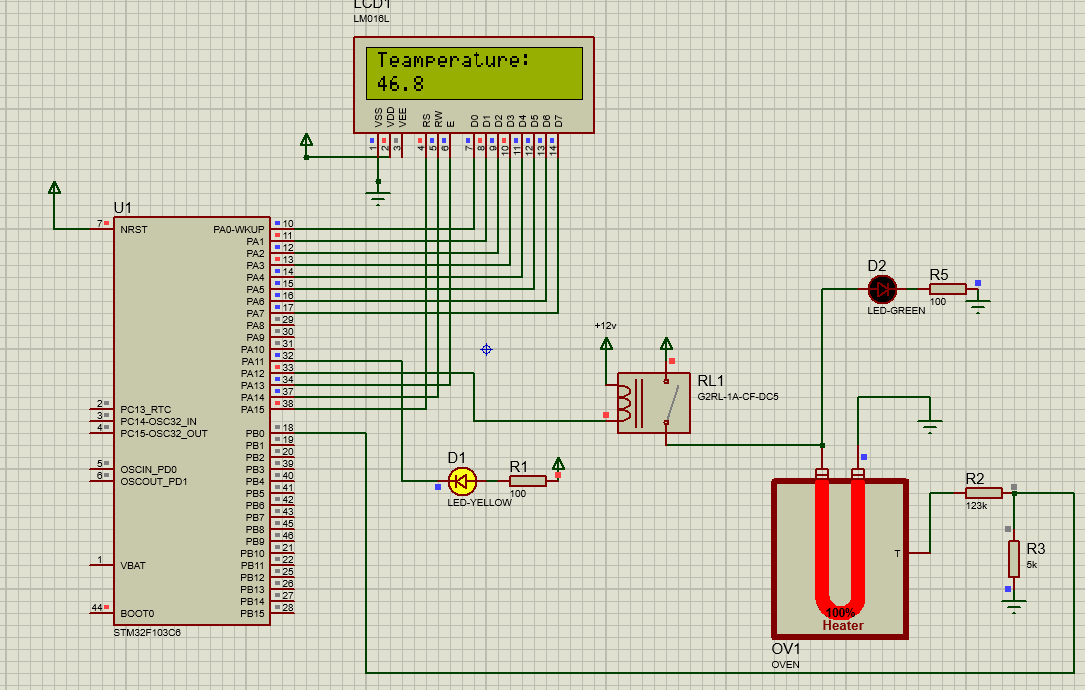
继电器的控制端与微控制器的PA12连接。继电器的另外一个控制端接上12v的电源。

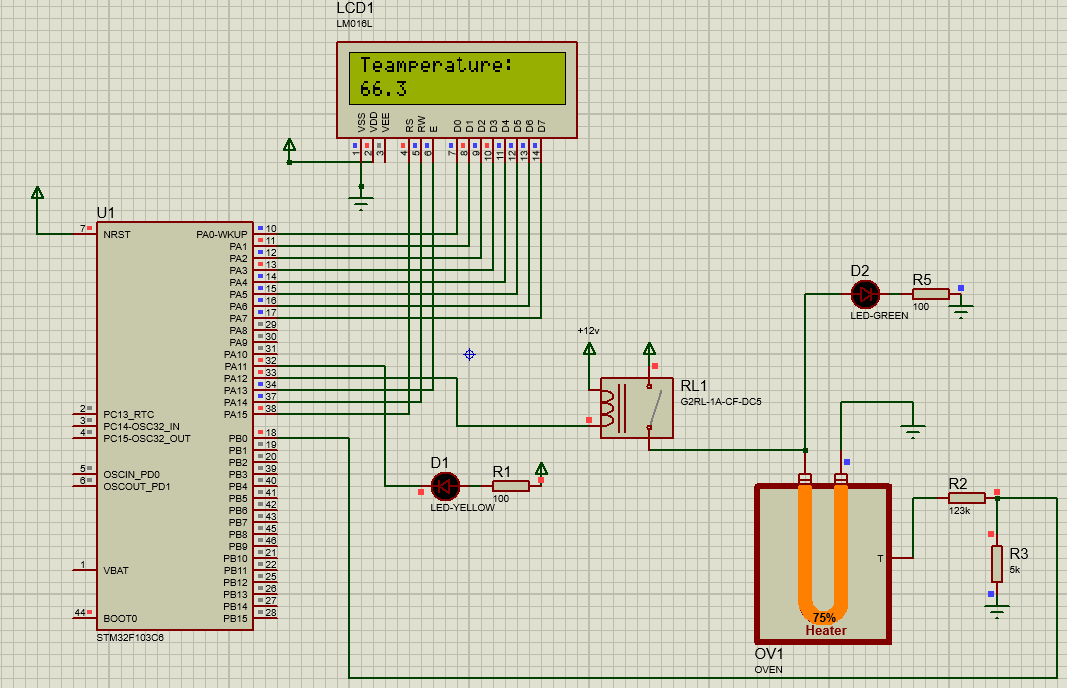


三、仿真结果展示

首先，点击仿真，Oven的Heater开始缓慢上升，温度从25°开始，电炉一直运行，温度传感器测的温度慢慢上升，D1每变换一次，温度显示器更新一下温度，到达预定温度（40°）D2灯灭，但是电炉有余温，温度还会一直上升，直到电炉Heater降为0%。







四、参考文献

[1] 温茹涵. 基于STM32的模糊PID温度控制系统[D].

[2] ybhuangfugui. Keil（MDK-ARM）介绍、下载、安装与注册 [EB/OL].https://blog.csdn.net/ybhuangfugui/article/details/51501781, 2016-05-25.

[3] SunWorshipper99. Protues8.8安装包、详细破解教程、安装常见问题解决办法总结[EB/OL]. https://blog.csdn.net/sunlight\_90/article/details/90274333. 2019-05-16

五、程序代码

1. /\* USER CODE BEGIN Header \*/
2. /\*\*
3. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
4. \* @file           : main.c
5. \* @brief          : Main program body
6. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
7. \* @attention
8. \*
9. \* <h2><center>© Copyright (c) 2020 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 \*/
21. /\* Includes ------------------------------------------------------------------\*/
22. #include "main.h"
24. /\* Private includes ----------------------------------------------------------\*/
25. /\* USER CODE BEGIN Includes \*/
26. #include "stdio.h"
27. #include "string.h"
28. /\* USER CODE END Includes \*/
30. /\* Private typedef -----------------------------------------------------------\*/
31. /\* USER CODE BEGIN PTD \*/
33. /\* USER CODE END PTD \*/
35. /\* Private define ------------------------------------------------------------\*/
36. /\* USER CODE BEGIN PD \*/
38. /\* USER CODE END PD \*/
40. /\* Private macro -------------------------------------------------------------\*/
41. /\* USER CODE BEGIN PM \*/
43. /\* USER CODE END PM \*/
45. /\* Private variables ---------------------------------------------------------\*/
46. ADC\_HandleTypeDef hadc1;
48. TIM\_HandleTypeDef htim3;
50. /\* USER CODE BEGIN PV \*/
51. uint8\_t **const** table1[] = "Teamperature:";
52. **char** buff[10];
53. /\* USER CODE END PV \*/
55. /\* Private function prototypes -----------------------------------------------\*/
56. **void** SystemClock\_Config(**void**);
57. **static** **void** MX\_GPIO\_Init(**void**);
58. **static** **void** MX\_ADC1\_Init(**void**);
59. **static** **void** MX\_TIM3\_Init(**void**);
60. /\* USER CODE BEGIN PFP \*/
62. /\* USER CODE END PFP \*/
64. /\* Private user code ---------------------------------------------------------\*/
65. /\* USER CODE BEGIN 0 \*/
66. **void** printFloat(**float** value){
67. **int** tmp,tmp1;
68. tmp = (**int**)value;
69. tmp1 = (**int**)((value-tmp)\*10)%10;
70. sprintf(&buff[0],"%d.%d\r\n",tmp,tmp1);
71. }
72. **void** Delay\_us(**int** us){
73. unsigned **short** differ = 0xffff-us-5;
74. \_\_HAL\_TIM\_SET\_COUNTER(&htim3,differ);
75. HAL\_TIM\_Base\_Start(&htim3);
76. **while**(differ<0xffff-6)
77. differ = \_\_HAL\_TIM\_GET\_COUNTER(&htim3);
78. HAL\_TIM\_Base\_Stop(&htim3);
80. }
81. **void** LcdWriteCom(uint8\_t com){
82. Delay\_us(20) ;
83. GPIOA->BSRR = 0x00ff0000;
84. GPIOA->BSRR = (com) ;
85. HAL\_GPIO\_WritePin (GPIOA,GPIO\_PIN\_15,GPIO\_PIN\_RESET) ;
86. HAL\_GPIO\_WritePin (GPIOA,GPIO\_PIN\_14,GPIO\_PIN\_RESET) ;
87. HAL\_GPIO\_WritePin (GPIOA,GPIO\_PIN\_13,GPIO\_PIN\_RESET) ;
88. Delay\_us(10) ;
89. HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_13,GPIO\_PIN\_SET) ;
90. Delay\_us(10) ;
91. HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_13,GPIO\_PIN\_RESET) ;
92. Delay\_us(10) ;
93. Delay\_us(20) ;
94. }
95. **void** LcdWriteDate (uint8\_t date){
96. Delay\_us(20) ;
98. GPIOA->BSRR = 0x00ff0000;
100. GPIOA->BSRR = (date) ;
102. HAL\_GPIO\_WritePin (GPIOA,GPIO\_PIN\_15,GPIO\_PIN\_SET) ;
103. HAL\_GPIO\_WritePin (GPIOA,GPIO\_PIN\_14,GPIO\_PIN\_RESET) ;
104. HAL\_GPIO\_WritePin (GPIOA,GPIO\_PIN\_13,GPIO\_PIN\_RESET) ;
105. Delay\_us (10) ;
106. HAL\_GPIO\_WritePin (GPIOA,GPIO\_PIN\_13,GPIO\_PIN\_SET) ;
107. Delay\_us(10) ;
108. HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_13,GPIO\_PIN\_RESET) ;
109. Delay\_us(10) ;
110. }
111. **void** LCD1602Init (**void**){
113. uint8\_t index=0;
115. HAL\_Delay(100) ;
117. LcdWriteCom(0x38); //???16\*2?????8????????cdWriteCom(0x0c); //?€???????????????
119. LcdWriteCom(0x0c);
120. LcdWriteCom (0x06);//?????????????€
122. LcdWriteCom(0x01) ;//???
124. HAL\_Delay(100) ;//????€????????????LCD1602???
125. LcdWriteCom(0x80);//?????????????€???
127. **for** (index=0; index<13; index++)
128. LcdWriteDate (table1 [index]); //????
130. }
131. **void** LCD1602WriteCommand(uint8\_t comm){
132. LcdWriteCom(0xc0+14);
133. LcdWriteDate(comm);
134. }
135. /\* USER CODE END 0 \*/
137. /\*\*
138. \* @brief  The application entry point.
139. \* @retval int
140. \*/
141. **int** main(**void**)
142. {
143. /\* USER CODE BEGIN 1 \*/
144. uint16\_t adc\_v;
145. uint8\_t i;
146. /\* USER CODE END 1 \*/

149. /\* MCU Configuration--------------------------------------------------------\*/
151. /\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/
152. HAL\_Init();
154. /\* USER CODE BEGIN Init \*/
156. /\* USER CODE END Init \*/
158. /\* Configure the system clock \*/
159. SystemClock\_Config();
161. /\* USER CODE BEGIN SysInit \*/
163. /\* USER CODE END SysInit \*/
165. /\* Initialize all configured peripherals \*/
166. MX\_GPIO\_Init();
167. MX\_ADC1\_Init();
168. MX\_TIM3\_Init();
169. /\* USER CODE BEGIN 2 \*/
170. HAL\_ADCEx\_Calibration\_Start(&hadc1);
171. LCD1602Init();
172. HAL\_Delay(10);
173. /\* USER CODE END 2 \*/
175. /\* Infinite loop \*/
176. /\* USER CODE BEGIN WHILE \*/
177. **while** (1)
178. {
179. /\* USER CODE END WHILE \*/
181. /\* USER CODE BEGIN 3 \*/
182. HAL\_GPIO\_TogglePin(GPIOA,GPIO\_PIN\_11);
183. HAL\_ADC\_Start(&hadc1);
184. HAL\_ADC\_PollForConversion(&hadc1,50);
185. **if**(HAL\_IS\_BIT\_SET(HAL\_ADC\_GetState(&hadc1),HAL\_ADC\_STATE\_REG\_EOC)){
186. adc\_v = HAL\_ADC\_GetValue(&hadc1);
187. printFloat((**float**)adc\_v\*128/4096);
188. LcdWriteCom(0xc0);
189. **for**(i =0;i<strlen(buff);i++){
190. LcdWriteDate(buff[i]);
191. }
192. }
193. **int** mypos = 40;//我的学号46，预定温度35+（4+6）/2 = 40
194. **if** ((**float**)adc\_v\*128/4096>mypos){
195. HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_12,GPIO\_PIN\_SET);
196. }**else**
197. HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_12,GPIO\_PIN\_RESET);
199. HAL\_Delay(500);
200. }
201. /\* USER CODE END 3 \*/
202. }
204. /\*\*
205. \* @brief System Clock Configuration
206. \* @retval None
207. \*/
208. **void** SystemClock\_Config(**void**)
209. {
210. RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};
211. RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};
212. RCC\_PeriphCLKInitTypeDef PeriphClkInit = {0};
214. /\*\* Initializes the CPU, AHB and APB busses clocks
215. \*/
216. RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;
217. RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;
218. RCC\_OscInitStruct.HSEPredivValue = RCC\_HSE\_PREDIV\_DIV1;
219. RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;
220. RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;
221. RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;
222. RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL9;
223. **if** (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)
224. {
225. Error\_Handler();
226. }
227. /\*\* Initializes the CPU, AHB and APB busses clocks
228. \*/
229. RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK
230. |RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;
231. RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;
232. RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;
233. RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;
234. RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;
236. **if** (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != HAL\_OK)
237. {
238. Error\_Handler();
239. }
240. PeriphClkInit.PeriphClockSelection = RCC\_PERIPHCLK\_ADC;
241. PeriphClkInit.AdcClockSelection = RCC\_ADCPCLK2\_DIV6;
242. **if** (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInit) != HAL\_OK)
243. {
244. Error\_Handler();
245. }
246. }
248. /\*\*
249. \* @brief ADC1 Initialization Function
250. \* @param None
251. \* @retval None
252. \*/
253. **static** **void** MX\_ADC1\_Init(**void**)
254. {
256. /\* USER CODE BEGIN ADC1\_Init 0 \*/
258. /\* USER CODE END ADC1\_Init 0 \*/
260. ADC\_ChannelConfTypeDef sConfig = {0};
262. /\* USER CODE BEGIN ADC1\_Init 1 \*/
264. /\* USER CODE END ADC1\_Init 1 \*/
265. /\*\* Common config
266. \*/
267. hadc1.Instance = ADC1;
268. hadc1.Init.ScanConvMode = ADC\_SCAN\_DISABLE;
269. hadc1.Init.ContinuousConvMode = DISABLE;
270. hadc1.Init.DiscontinuousConvMode = DISABLE;
271. hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;
272. hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;
273. hadc1.Init.NbrOfConversion = 1;
274. **if** (HAL\_ADC\_Init(&hadc1) != HAL\_OK)
275. {
276. Error\_Handler();
277. }
278. /\*\* Configure Regular Channel
279. \*/
280. sConfig.Channel = ADC\_CHANNEL\_8;
281. sConfig.Rank = ADC\_REGULAR\_RANK\_1;
282. sConfig.SamplingTime = ADC\_SAMPLETIME\_239CYCLES\_5;
283. **if** (HAL\_ADC\_ConfigChannel(&hadc1, &sConfig) != HAL\_OK)
284. {
285. Error\_Handler();
286. }
287. /\* USER CODE BEGIN ADC1\_Init 2 \*/
289. /\* USER CODE END ADC1\_Init 2 \*/
291. }
293. /\*\*
294. \* @brief TIM3 Initialization Function
295. \* @param None
296. \* @retval None
297. \*/
298. **static** **void** MX\_TIM3\_Init(**void**)
299. {
301. /\* USER CODE BEGIN TIM3\_Init 0 \*/
303. /\* USER CODE END TIM3\_Init 0 \*/
305. TIM\_ClockConfigTypeDef sClockSourceConfig = {0};
306. TIM\_MasterConfigTypeDef sMasterConfig = {0};
308. /\* USER CODE BEGIN TIM3\_Init 1 \*/
310. /\* USER CODE END TIM3\_Init 1 \*/
311. htim3.Instance = TIM3;
312. htim3.Init.Prescaler = 72-1;
313. htim3.Init.CounterMode = TIM\_COUNTERMODE\_UP;
314. htim3.Init.Period = 5000;
315. htim3.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;
316. htim3.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;
317. **if** (HAL\_TIM\_Base\_Init(&htim3) != HAL\_OK)
318. {
319. Error\_Handler();
320. }
321. sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;
322. **if** (HAL\_TIM\_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL\_OK)
323. {
324. Error\_Handler();
325. }
326. sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;
327. sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;
328. **if** (HAL\_TIMEx\_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL\_OK)
329. {
330. Error\_Handler();
331. }
332. /\* USER CODE BEGIN TIM3\_Init 2 \*/
334. /\* USER CODE END TIM3\_Init 2 \*/
336. }
338. /\*\*
339. \* @brief GPIO Initialization Function
340. \* @param None
341. \* @retval None
342. \*/
343. **static** **void** MX\_GPIO\_Init(**void**)
344. {
345. GPIO\_InitTypeDef GPIO\_InitStruct = {0};
347. /\* GPIO Ports Clock Enable \*/
348. \_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();
349. \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();
350. \_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();
352. /\*Configure GPIO pin Output Level \*/
353. HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3
354. |GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6|GPIO\_PIN\_7
355. |GPIO\_PIN\_11|GPIO\_PIN\_12|GPIO\_PIN\_13|GPIO\_PIN\_14
356. |GPIO\_PIN\_15, GPIO\_PIN\_RESET);
358. /\*Configure GPIO pins : PA0 PA1 PA2 PA3
359. PA4 PA5 PA6 PA7
360. PA11 PA12 PA13 PA14
361. PA15 \*/
362. GPIO\_InitStruct.Pin = GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3
363. |GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6|GPIO\_PIN\_7
364. |GPIO\_PIN\_11|GPIO\_PIN\_12|GPIO\_PIN\_13|GPIO\_PIN\_14
365. |GPIO\_PIN\_15;
366. GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;
367. GPIO\_InitStruct.Pull = GPIO\_NOPULL;
368. GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;
369. HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);
371. }
373. /\* USER CODE BEGIN 4 \*/
375. /\* USER CODE END 4 \*/
377. /\*\*
378. \* @brief  This function is executed in case of error occurrence.
379. \* @retval None
380. \*/
381. **void** Error\_Handler(**void**)
382. {
383. /\* USER CODE BEGIN Error\_Handler\_Debug \*/
384. /\* User can add his own implementation to report the HAL error return state \*/
386. /\* USER CODE END Error\_Handler\_Debug \*/
387. }
389. #ifdef  USE\_FULL\_ASSERT
390. /\*\*
391. \* @brief  Reports the name of the source file and the source line number
392. \*         where the assert\_param error has occurred.
393. \* @param  file: pointer to the source file name
394. \* @param  line: assert\_param error line source number
395. \* @retval None
396. \*/
397. **void** assert\_failed(uint8\_t \*file, uint32\_t line)
398. {
399. /\* USER CODE BEGIN 6 \*/
400. /\* User can add his own implementation to report the file name and line number,
401. tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/
402. /\* USER CODE END 6 \*/
403. }
404. #endif /\* USE\_FULL\_ASSERT \*/
406. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/