1. */\* LEDC (LED Controller) fade example*
2. This example code is in the Public Domain (or CC0 licensed, at your option.)
3. Unless required by applicable law or agreed to in writing, this
4. software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
5. CONDITIONS OF ANY KIND, either express or implied.
6. \*/
7. #include <stdio.h>
8. #include "freertos/FreeRTOS.h"
9. #include "freertos/task.h"
10. #include "driver/ledc.h"
11. #include "esp\_err.h"
12. #include "freertos/FreeRTOS.h"
13. #include "freertos/semphr.h"
14. */\**
15. \* About this example
16. \*
17. \* 1. Start with initializing LEDC module:
18. \*    a. Set the timer of LEDC first, this determines the frequency
19. \*       and resolution of PWM.
20. \*    b. Then set the LEDC channel you want to use,
21. \*       and bind with one of the timers.
22. \*
23. \* 2. You need first to install a default fade function,
24. \*    then you can use fade APIs.
25. \*
26. \* 3. You can also set a target duty directly without fading.
27. \*
28. \* 4. This example uses GPIO18/19/4/5 as LEDC output,
29. \*    and it will change the duty repeatedly.
30. \*
31. \* 5. GPIO18/19 are from high speed channel group.
32. \*    GPIO4/5 are from low speed channel group.
33. \*
34. \*/
35. #if CONFIG\_IDF\_TARGET\_ESP32
36. #define LEDC\_HS\_TIMER          LEDC\_TIMER\_0
37. #define LEDC\_HS\_MODE           LEDC\_HIGH\_SPEED\_MODE
38. #define LEDC\_HS\_CH0\_GPIO       (18)
39. #define LEDC\_HS\_CH0\_CHANNEL    LEDC\_CHANNEL\_0
40. #define LEDC\_HS\_CH1\_GPIO       (19)
41. #define LEDC\_HS\_CH1\_CHANNEL    LEDC\_CHANNEL\_1
42. #endif
43. #define LEDC\_LS\_TIMER          LEDC\_TIMER\_1
44. #define LEDC\_LS\_MODE           LEDC\_LOW\_SPEED\_MODE
45. #if !CONFIG\_IDF\_TARGET\_ESP32
46. #define LEDC\_LS\_CH0\_GPIO       (18)
47. #define LEDC\_LS\_CH0\_CHANNEL    LEDC\_CHANNEL\_0
48. #define LEDC\_LS\_CH1\_GPIO       (19)
49. #define LEDC\_LS\_CH1\_CHANNEL    LEDC\_CHANNEL\_1
50. #endif
51. #define LEDC\_LS\_CH2\_GPIO       (6)
52. #define LEDC\_LS\_CH2\_CHANNEL    LEDC\_CHANNEL\_2
53. #define LEDC\_LS\_CH3\_GPIO       (5)
54. #define LEDC\_LS\_CH3\_CHANNEL    LEDC\_CHANNEL\_3
55. #define LEDC\_TEST\_CH\_NUM       (4)
56. #define LEDC\_TEST\_DUTY         (4000)
57. #define LEDC\_TEST\_FADE\_TIME    (3000)
58. */\**
59. \* This callback function will be called when fade operation has ended
60. \* Use callback only if you are aware it is being called inside an ISR
61. \* Otherwise, you can use a semaphore to unblock tasks
62. \*/
63. static bool cb\_ledc\_fade\_end\_event(const ledc\_cb\_param\_t \*param, void \*user\_arg)
64. {
65. portBASE\_TYPE taskAwoken = pdFALSE;
66. if (param->event == LEDC\_FADE\_END\_EVT) {
67. SemaphoreHandle\_t counting\_sem = (SemaphoreHandle\_t) user\_arg;
68. xSemaphoreGiveFromISR(counting\_sem, &taskAwoken);
69. }
70. return (taskAwoken == pdTRUE);
71. }
72. void app\_main(void)
73. {
74. int ch;
75. */\**
76. \* Prepare and set configuration of timers
77. \* that will be used by LED Controller
78. \*/
79. ledc\_timer\_config\_t ledc\_timer = {
80. .duty\_resolution = LEDC\_TIMER\_13\_BIT, *// resolution of PWM duty*
81. .freq\_hz = 5000,                      *// frequency of PWM signal*
82. .speed\_mode = LEDC\_LS\_MODE,           *// timer mode*
83. .timer\_num = LEDC\_LS\_TIMER,            *// timer index*
84. .clk\_cfg = LEDC\_AUTO\_CLK,              *// Auto select the source clock*
85. };
86. *// Set configuration of timer0 for high speed channels*
87. ledc\_timer\_config(&ledc\_timer);
88. #ifdef CONFIG\_IDF\_TARGET\_ESP32
89. *// Prepare and set configuration of timer1 for low speed channels*
90. ledc\_timer.speed\_mode = LEDC\_HS\_MODE;
91. ledc\_timer.timer\_num = LEDC\_HS\_TIMER;
92. ledc\_timer\_config(&ledc\_timer);
93. #endif
94. */\**
95. \* Prepare individual configuration
96. \* for each channel of LED Controller
97. \* by selecting:
98. \* - controller's channel number
99. \* - output duty cycle, set initially to 0
100. \* - GPIO number where LED is connected to
101. \* - speed mode, either high or low
102. \* - timer servicing selected channel
103. \*   Note: if different channels use one timer,
104. \*         then frequency and bit\_num of these channels
105. \*         will be the same
106. \*/
107. ledc\_channel\_config\_t ledc\_channel[LEDC\_TEST\_CH\_NUM] = {
108. #if CONFIG\_IDF\_TARGET\_ESP32
109. {
110. .channel    = LEDC\_HS\_CH0\_CHANNEL,
111. .duty       = 0,
112. .gpio\_num   = LEDC\_HS\_CH0\_GPIO,
113. .speed\_mode = LEDC\_HS\_MODE,
114. .hpoint     = 0,
115. .timer\_sel  = LEDC\_HS\_TIMER,
116. .flags.output\_invert = 0
117. },
118. {
119. .channel    = LEDC\_HS\_CH1\_CHANNEL,
120. .duty       = 0,
121. .gpio\_num   = LEDC\_HS\_CH1\_GPIO,
122. .speed\_mode = LEDC\_HS\_MODE,
123. .hpoint     = 0,
124. .timer\_sel  = LEDC\_HS\_TIMER,
125. .flags.output\_invert = 0
126. },
127. #else
128. {
129. .channel    = LEDC\_LS\_CH0\_CHANNEL,
130. .duty       = 0,
131. .gpio\_num   = LEDC\_LS\_CH0\_GPIO,
132. .speed\_mode = LEDC\_LS\_MODE,
133. .hpoint     = 0,
134. .timer\_sel  = LEDC\_LS\_TIMER,
135. .flags.output\_invert = 0
136. },
137. {
138. .channel    = LEDC\_LS\_CH1\_CHANNEL,
139. .duty       = 0,
140. .gpio\_num   = LEDC\_LS\_CH1\_GPIO,
141. .speed\_mode = LEDC\_LS\_MODE,
142. .hpoint     = 0,
143. .timer\_sel  = LEDC\_LS\_TIMER,
144. .flags.output\_invert = 0
145. },
146. #endif
147. {
148. .channel    = LEDC\_LS\_CH2\_CHANNEL,
149. .duty       = 0,
150. .gpio\_num   = LEDC\_LS\_CH2\_GPIO,
151. .speed\_mode = LEDC\_LS\_MODE,
152. .hpoint     = 0,
153. .timer\_sel  = LEDC\_LS\_TIMER,
154. .flags.output\_invert = 1
155. },
156. {
157. .channel    = LEDC\_LS\_CH3\_CHANNEL,
158. .duty       = 0,
159. .gpio\_num   = LEDC\_LS\_CH3\_GPIO,
160. .speed\_mode = LEDC\_LS\_MODE,
161. .hpoint     = 0,
162. .timer\_sel  = LEDC\_LS\_TIMER,
163. .flags.output\_invert = 1
164. },
165. };
166. *// Set LED Controller with previously prepared configuration*
167. for (ch = 0; ch < LEDC\_TEST\_CH\_NUM; ch++) {
168. ledc\_channel\_config(&ledc\_channel[ch]);
169. }
170. *// Initialize fade service.*
171. ledc\_fade\_func\_install(0);
172. ledc\_cbs\_t callbacks = {
173. .fade\_cb = cb\_ledc\_fade\_end\_event
174. };
175. SemaphoreHandle\_t counting\_sem = xSemaphoreCreateCounting(LEDC\_TEST\_CH\_NUM, 0);
176. for (ch = 0; ch < LEDC\_TEST\_CH\_NUM; ch++) {
177. ledc\_cb\_register(ledc\_channel[ch].speed\_mode, ledc\_channel[ch].channel, &callbacks, (void \*) counting\_sem);
178. }
179. while (1) {
180. printf("1. LEDC fade up to duty = %d\n", LEDC\_TEST\_DUTY);
181. for (ch = 0; ch < LEDC\_TEST\_CH\_NUM; ch++) {
182. ledc\_set\_fade\_with\_time(ledc\_channel[ch].speed\_mode,
183. ledc\_channel[ch].channel, LEDC\_TEST\_DUTY, LEDC\_TEST\_FADE\_TIME);
184. ledc\_fade\_start(ledc\_channel[ch].speed\_mode,
185. ledc\_channel[ch].channel, LEDC\_FADE\_NO\_WAIT);
186. }
187. for (int i = 0; i < LEDC\_TEST\_CH\_NUM; i++) {
188. xSemaphoreTake(counting\_sem, portMAX\_DELAY);
189. }
190. printf("2. LEDC fade down to duty = 0\n");
191. for (ch = 0; ch < LEDC\_TEST\_CH\_NUM; ch++) {
192. ledc\_set\_fade\_with\_time(ledc\_channel[ch].speed\_mode,
193. ledc\_channel[ch].channel, 0, LEDC\_TEST\_FADE\_TIME);
194. ledc\_fade\_start(ledc\_channel[ch].speed\_mode,
195. ledc\_channel[ch].channel, LEDC\_FADE\_NO\_WAIT);
196. }
197. for (int i = 0; i < LEDC\_TEST\_CH\_NUM; i++) {
198. xSemaphoreTake(counting\_sem, portMAX\_DELAY);
199. }
200. printf("3. LEDC set duty = %d without fade\n", LEDC\_TEST\_DUTY);
201. for (ch = 0; ch < LEDC\_TEST\_CH\_NUM; ch++) {
202. ledc\_set\_duty(ledc\_channel[ch].speed\_mode, ledc\_channel[ch].channel, LEDC\_TEST\_DUTY);
203. ledc\_update\_duty(ledc\_channel[ch].speed\_mode, ledc\_channel[ch].channel);
204. }
205. vTaskDelay(1000 / portTICK\_PERIOD\_MS);
206. printf("4. LEDC set duty = 0 without fade\n");
207. for (ch = 0; ch < LEDC\_TEST\_CH\_NUM; ch++) {
208. ledc\_set\_duty(ledc\_channel[ch].speed\_mode, ledc\_channel[ch].channel, 0);
209. ledc\_update\_duty(ledc\_channel[ch].speed\_mode, ledc\_channel[ch].channel);
210. }
211. vTaskDelay(1000 / portTICK\_PERIOD\_MS);
212. }
213. }