Lab7 code

1. //pingpong template file for Lab #7
2. //Revised by Jianjian Song to add pressing early penalty
3. //June 2019
4. #include "xparameters.h"
5. #include "xgpio.h"
6. #include "led\_ip.h"
7. // Include scutimer header file
8. #include "XScuTimer.h"
9. //====================================================
10. XScuTimer Timer; /\* Cortex A9 SCU Private Timer Instance \*/
11. **int** delay(**int** i);
12. **void** MoveBallRight(**void**);
13. **void** MoveBallLeft(**void**);
15. #define ONE\_TENTH 32500000 // half of the CPU clock speed/10
16. #define START 1
17. #define STOP 0
18. #define LEFT 0
19. #define RIGHT 1
20. #define RESETBUTTON 0b0100
21. #define STARTBUTTON 0b0010
22. #define LEFTPADDLE 0b1000
23. #define RIGHTPADDLE 0b0001
25. #define LED\_PATTERNS\_ORDER\_LEFT\_OUT -1
26. #define LED\_PATTERNS\_ORDER\_RIGHT\_OUT 6
28. #define RUNKEEP 1
29. #define RUNFOBBIDEN 0
31. **int** psb\_check, dip\_check, LedState, Status, dip\_check\_prev, psb\_check\_prev;
33. XGpio dip, push;
35. // PS Timer related definitions
36. XScuTimer\_Config \*ConfigPtr;
37. XScuTimer \*TimerInstancePtr = &Timer;
39. **int** LED\_PATTERNS[6] = { 0b0000, 0b1000, 0b0100, 0b0010, 0b0001, 0b0000 };
40. **int** LED\_PATTERNS\_ORDER[6] = {0,1,2,3,4,5};
41. **int** scoreright, scoreleft;
42. **char** GameOver, StartDirection;
44. **int** main(**void**) {
46. //initialize variables, timers, ports
47. XGpio\_Initialize(&dip, XPAR\_SWITCHES\_DEVICE\_ID);
48. XGpio\_SetDataDirection(&dip, 1, 0xffffffff);
50. XGpio\_Initialize(&push, XPAR\_BOTTONS\_DEVICE\_ID);
51. XGpio\_SetDataDirection(&push, 1, 0xffffffff);
53. //use psb\_check\_prev and psb\_check to check if the button has been push
54. psb\_check\_prev = XGpio\_DiscreteRead(&push, 1);
56. // Initialize the timer
57. ConfigPtr = XScuTimer\_LookupConfig(XPAR\_PS7\_SCUTIMER\_0\_DEVICE\_ID);
58. Status = XScuTimer\_CfgInitialize(TimerInstancePtr, ConfigPtr,
59. ConfigPtr->BaseAddr);
60. **if** (Status != XST\_SUCCESS) {
61. xil\_printf("Timer init() failed\r\n");
62. **return** XST\_FAILURE;
63. }
65. // Read dip switch values
66. dip\_check\_prev = XGpio\_DiscreteRead(&dip, 1);
67. // Load timer with delay in multiple of ONE\_TENTH
68. XScuTimer\_LoadTimer(TimerInstancePtr, ONE\_TENTH\*dip\_check\_prev);
69. // Set AutoLoad mode
70. XScuTimer\_EnableAutoReload(TimerInstancePtr);
71. // Start the timer
72. XScuTimer\_Start(TimerInstancePtr);
74. xil\_printf("-- Start of the Ping Pong Program --\r\n");
75. GameOver = STOP;
76. scoreright = 0;
77. scoreleft = 0;
78. xil\_printf("Score Left = %d   Score Right = %d\r\n", scoreright, scoreleft);
79. StartDirection = LEFT;
80. **while** (1) {
81. // Read push buttons and reset score if Button 2 is pressed
82. psb\_check = XGpio\_DiscreteRead(&push, 1);
83. **if** (psb\_check == RESETBUTTON && psb\_check != psb\_check\_prev)//reset game
84. {
85. xil\_printf("\n\rNew Game - Scores Reset\r\n");
86. scoreright = 0;
87. scoreleft = 0;
88. xil\_printf("Score Left = %d   Score Right = %d\r\n", scoreright,
89. scoreleft);
90. psb\_check\_prev = psb\_check;
91. }
93. //check the STARTBUTTON been pushed
94. **if** (psb\_check == STARTBUTTON) {
95. GameOver = START;   //start game
96. psb\_check\_prev = psb\_check;
97. }
99. //check the game status,if start the ball will move right or right
100. **if** (GameOver == STOP) {
101. **if** (StartDirection == LEFT) {
102. LED\_IP\_mWriteReg(XPAR\_LED\_IP\_0\_S00\_AXI\_BASEADDR, 0,
103. LED\_PATTERNS[5]);
104. } **else** {
105. LED\_IP\_mWriteReg(XPAR\_LED\_IP\_0\_S00\_AXI\_BASEADDR, 0,
106. LED\_PATTERNS[0]);
107. }
108. } **else** {
109. **if** (StartDirection) {
110. MoveBallLeft();
111. } **else** {
112. MoveBallRight();
113. }
114. }

117. }
118. } //main()
120. **void** MoveBallRight(**void**) {
121. **int** led\_order;
122. **int** run;
124. //keep the score add once
125. **int** getOut = 1;
127. **for** (led\_order = LED\_PATTERNS\_ORDER[5]; led\_order >= LED\_PATTERNS\_ORDER[0];) {
129. run = RUNKEEP;
130. **while** (run) {
131. //check the switches changed
132. dip\_check = XGpio\_DiscreteRead(&dip, 1);
133. **if** (dip\_check != dip\_check\_prev) {
134. xil\_printf("DIP Switch Status %x, %x\r\n", dip\_check\_prev,
135. dip\_check);
136. dip\_check\_prev = dip\_check;
137. // load timer with the new switch settings
138. XScuTimer\_LoadTimer(TimerInstancePtr, ONE\_TENTH\*dip\_check);
139. }
141. //check the LEFTPADDLE pushed
142. psb\_check = XGpio\_DiscreteRead(&push, 1);
143. **if** (psb\_check == LEFTPADDLE) {
144. //set StartDirection
145. **if** (led\_order == LED\_PATTERNS\_ORDER[1]) {
146. StartDirection = RIGHT;
147. run = RUNFOBBIDEN;
148. led\_order = LED\_PATTERNS\_ORDER\_LEFT\_OUT;
149. } **else** {
150. GameOver = STOP;
151. LED\_IP\_mWriteReg(XPAR\_LED\_IP\_0\_S00\_AXI\_BASEADDR, 0,
152. LED\_PATTERNS[0]);
153. StartDirection = RIGHT;
154. scoreleft += 1;
155. run = RUNFOBBIDEN;
156. led\_order = LED\_PATTERNS\_ORDER\_LEFT\_OUT;
157. LED\_IP\_mWriteReg(XPAR\_LED\_IP\_0\_S00\_AXI\_BASEADDR, 0,
158. LED\_PATTERNS[led\_order]);
159. xil\_printf("Score Left = %d   Score Right = %d\r\n", scoreright, scoreleft);
160. }
161. } **else** {
162. //set GameOver; display scores
163. **if** (led\_order == LED\_PATTERNS\_ORDER[0] && getOut) {
164. scoreleft += 1;
165. GameOver = STOP;
166. StartDirection = RIGHT;
168. xil\_printf("Score Left = %d   Score Right = %d\r\n",
169. scoreright, scoreleft);
170. run = RUNFOBBIDEN;
171. getOut = 0;
172. }
174. **if** (XScuTimer\_IsExpired(TimerInstancePtr)) {
175. // clear status bit
176. XScuTimer\_ClearInterruptStatus(TimerInstancePtr);
177. led\_order--;
178. LED\_IP\_mWriteReg(XPAR\_LED\_IP\_0\_S00\_AXI\_BASEADDR, 0,
179. LED\_PATTERNS[led\_order]);
180. run = RUNFOBBIDEN;
181. }
182. }
183. }
185. }
186. }
188. **void** MoveBallLeft(**void**) {
189. **int** led\_order;
190. **int** run;
192. //keep the score add once
193. **int** getOut = 1;
194. **for** (led\_order = LED\_PATTERNS\_ORDER[1]; led\_order <= LED\_PATTERNS\_ORDER[5];) {
195. run = RUNKEEP;
196. **while** (run) {
197. dip\_check = XGpio\_DiscreteRead(&dip, 1);
198. **if** (dip\_check != dip\_check\_prev) {
199. xil\_printf("DIP Switch Status %x, %x\r\n", dip\_check\_prev,
200. dip\_check);
201. dip\_check\_prev = dip\_check;
202. // load timer with the new switch settings
203. XScuTimer\_LoadTimer(TimerInstancePtr, ONE\_TENTH\*dip\_check);
204. }
205. psb\_check = XGpio\_DiscreteRead(&push, 1);
206. **if** (psb\_check == RIGHTPADDLE) {
207. //set StartDirection
208. **if** (led\_order == LED\_PATTERNS\_ORDER[4]) {
209. StartDirection = LEFT;
210. run = 0;
211. led\_order = LED\_PATTERNS\_ORDER\_RIGHT\_OUT;
212. } **else** {
213. GameOver = STOP;
214. StartDirection = LEFT;
215. scoreright += 1;
216. run = RUNFOBBIDEN;
218. led\_order = LED\_PATTERNS\_ORDER\_RIGHT\_OUT;
219. LED\_IP\_mWriteReg(XPAR\_LED\_IP\_0\_S00\_AXI\_BASEADDR, 0,
220. LED\_PATTERNS[led\_order]);
222. xil\_printf("Score Left = %d   Score Right = %d\r\n", scoreright, scoreleft);
223. }
225. } **else** {
226. **if** (led\_order == LED\_PATTERNS\_ORDER[5] && getOut) {
227. //set GameOver; display scores
228. scoreright += 1;
229. GameOver = STOP;
230. StartDirection = LEFT;
231. xil\_printf("Score Left = %d   Score Right = %d\r\n",
232. scoreright, scoreleft);
233. run = RUNFOBBIDEN;
234. getOut = 0;
235. }
237. **if** (XScuTimer\_IsExpired(TimerInstancePtr)) {
238. // clear status bit
239. XScuTimer\_ClearInterruptStatus(TimerInstancePtr);
240. led\_order++;
241. LED\_IP\_mWriteReg(XPAR\_LED\_IP\_0\_S00\_AXI\_BASEADDR, 0,
242. LED\_PATTERNS[led\_order]);
243. run = RUNFOBBIDEN;
244. }
245. }
246. }
247. }
248. }