Министерство образования и науки Российской Федерации

ФГАОУ ВПО «Уральский федеральный университет

имени первого Президента России Б.Н.Ельцина»

Институт радиоэлектроники и информационных технологий – РтФ

Кафедра технологий и средств связи

ОТЧЕТ

по лабораторной работе №5

**«Работа с АЦП»**

по дисциплине «Техника микропроцессорных  
систем в коммутации»

Группа РИ-430801

Студент Ахтаров Д.Н.

Найков А.В.

Преподаватель Гусев А.В.

Екатеринбург 2016

# Цель работы

Научиться работать с АЦП.

# Алгоритм

В программе представлен алгоритм популярной игры от радик геймс – змейка.

Были созданы объекты еда и змейка. Управление происходила с помощью переменной enum – перечисления направлений. Случайное положение еды задаётся собственным алгоритмом. Создан объект rand и ему присваиваются значения по x и y, зависящие от времени. При совпадении координаты головы змейки с координатой еды длина змейки увеличивается, а еда меняет своё положение. Изменение скорости змейки происходит с помощью АЦП. Игра заканчивается, если змейка врезается сама в себя, далее можно продолжить игру заново, нажав на джойстик.

Задаем настройки АЦП в функции initADC(), дальше в цикле while считываем значение АЦП и по линейному закону преобразуем скорость изменения координат змейки.

# Код программы

1. #include <LPC17xx.H>
2. #include "GLCD.h"
3. #include "LibraryDisplay.h"
5. **unsigned int i0, tick=0;**
6. unsigned int A[8];
7. unsigned int i=0, djostic, button, value;
8. unsigned int timing = 1100000, led = 0, step, hody = 0, score;
10. **struct Object {**
11. unsigned int x;
12. unsigned int y;
13. };
14. struct Object zmeyaDel;
15. **struct Object eda;**
16. struct Object rand;
18. struct MasObject {
19. unsigned int x[1024];
20. **unsigned int y[1024];**
21. unsigned int dlina;
22. };
23. struct MasObject zmeya;
25. **enum Vector {Up, Right, Down, Left};**
26. enum Vector vector = Up;

29. void SysTick\_Handler();
30. **void InitADC();**
31. void delay(int);
32. void port\_init();
33. void systick\_init();
34. void display\_color();
35. **void display\_line();**
36. void coord\_init();
37. void vector\_set(unsigned int);
38. void game\_over();
39. void init\_ShIM();
41. int main (void)
42. {
43. SystemInit ();
44. init\_display();
45. **port\_init();**
46. systick\_init();
47. InitADC();
48. display\_line();
49. display\_color();
50. **coord\_init();**
51. init\_ShIM();
53. while (1)
54. {
55. **timing = (value\*1000) + 300000;**
56. //GLCD\_DisplayU32(1, 1, 1, value);
57. if (led <= 4){
58. LPC\_GPIO2->FIOPIN = A[led];
59. LPC\_GPIO1->FIOPIN = 0;
60. **} else {**
61. LPC\_GPIO1->FIOPIN = A[led];
62. LPC\_GPIO2->FIOPIN = 0;
63. }
64. switch(vector)
65. **{**
66. case Up:
67. zmeya.y[hody] = zmeya.y[hody-1] - step;
68. zmeya.x[hody] = zmeya.x[hody-1];
69. break;
70. **case Right:**
71. zmeya.x[hody] = zmeya.x[hody-1] - step;
72. zmeya.y[hody] = zmeya.y[hody-1];
73. break;
74. case Down:
75. **zmeya.y[hody] = zmeya.y[hody-1] + step;**
76. zmeya.x[hody] = zmeya.x[hody-1];
77. break;
78. case Left:
79. zmeya.x[hody] = zmeya.x[hody-1] + step;
80. **zmeya.y[hody] = zmeya.y[hody-1];**
81. break;
82. }
84. if (zmeya.y[hody] > 225)
85. **zmeya.y[hody] = 10;**
86. else if (zmeya.y[hody] < 10)
87. zmeya.y[hody] = 225;
88. if (zmeya.x[hody] > 305)
89. zmeya.x[hody] = 10;
90. **else if (zmeya.x[hody] < 10)**
91. zmeya.x[hody] = 305;
93. if ((zmeya.x[hody] == eda.x) && (zmeya.y[hody] == eda.y))
94. {
95. **zmeya.dlina++;**
96. eda.x = rand.x;
97. eda.y = rand.y;
98. GLCD\_Bargraph(eda.x, eda.y, step, step, Purple);
99. }
101. GLCD\_Bargraph(zmeya.x[hody], zmeya.y[hody], step, step, Purple);
102. for (i=1; i<zmeya.dlina; i++)
103. {
104. if((zmeya.x[hody] == zmeya.x[hody-i]) && (zmeya.y[hody-i] == zmeya.y[hody]))
105. **{**
106. game\_over();
107. }
108. }
109. delay(timing);
110. **GLCD\_Bargraph(zmeya.x[hody-zmeya.dlina+1], zmeya.y[hody-zmeya.dlina+1], step, step, Blue);**
111. hody++;
112. if (hody > 1023)
113. {
114. hody = zmeya.dlina;
115. **for(i=zmeya.dlina; i > 0; i--)**
116. {
117. zmeya.x[zmeya.dlina-i] = zmeya.x[1024-i];
118. zmeya.y[zmeya.dlina-i] = zmeya.y[1024-i];
119. }
120. **}**
121. }
122. }
124. void SysTick\_Handler()
125. **{**
126. tick++;
127. if (tick >= 100)
128. {
129. tick = 0 ;
130. **rand.x+=step;**
131. rand.y+=step;
132. if (rand.y > 225)
133. rand.y = 10;
134. if (rand.x > 305)
135. **rand.x = 10;**
136. djostic = LPC\_GPIO1->FIOPIN;
137. vector\_set(djostic);
138. if((LPC\_ADC->ADGDR & (1<<31)) != 0)
139. {
140. **value = (LPC\_ADC->ADGDR >> 4) & 0x0FFF;**
141. LPC\_ADC->ADCR |= (1 << 24) | (1 << 2);
142. }
143. */\*button = (LPC\_GPIO2->FIOPIN)&(1<<10);*
144. *if (button == 0)*
145. ***{***
146. *timing -= 100000;*
147. *led++;*
148. *if (timing <= 300000)*
149. *{*
150. ***timing = 1100000;***
151. *led = 0;*
152. *}*
153. *}\*/*
154. }
155. **}**
157. void InitADC (void)
158. {
159. LPC\_PINCON->PINSEL1 = (1<<18);
160. **LPC\_ADC->ADCR = (1 << 2) |**
161. (0xf << 12) |
162. (0xf << 8) |
163. (1 << 21) |
164. (1 << 24) ;
165. **}**
167. void delay(int count){
168. for (i0 = 0; i0 < count; i0++);
169. }
171. void port\_init(){
172. A[0] = (1<<6);
173. A[1] = (1<<5);
174. A[2] = (1<<4);
175. **A[3] = (1<<3);**
176. A[4] = (1<<2);
177. A[5] = (1<<31);
178. A[6] = (1<<29);
179. A[7] = (1<<28);
180. **LPC\_GPIO2->FIODIR |= (1<<2) | (1<<3) | (1<<4) | (1<<5) | (1<<6); // P2.2 - P2.6 - OUT**
181. LPC\_GPIO1->FIODIR |= (1<<28) | (1<<29) | (1<<31); // P1.28, P1.29, P1.31 - OUT
182. }
184. void systick\_init(){
185. **SysTick->LOAD = 94000 - 1; */\* set reload register \*/***
186. SysTick->VAL = (0x00); */\* Load the SysTick Counter Value \*/*
187. SysTick->CTRL = (1 << 2)|(1<<0)|(1<<1);
188. }
190. **void display\_line(){**
191. GLCD\_Clear(White);
192. GLCD\_Bargraph(5, 5, 5, 230, Purple);
193. GLCD\_Bargraph(5, 5, 310, 5, Purple);
194. GLCD\_Bargraph(310, 5, 5, 230, Purple);
195. **GLCD\_Bargraph(5, 230, 310, 5, Purple);**
196. }
198. void display\_color(){
199. GLCD\_SetTextColor(Navy);
200. **//GLCD\_SetTextColor(Purple);**
201. }
203. void coord\_init()
204. {
205. **// for (i=1; i<1024; i++)**
206. // {
207. // zmeya.x[i] = 0;
208. // zmeya.y[i] = 0;
209. // }
210. **hody = 1;**
211. zmeya.x[0] = 150;
212. zmeya.y[0] = 110;
213. zmeya.dlina = 1;
215. **eda.x = 110;**
216. eda.y = 120;
218. rand.x = 10;
219. rand.y = 30;
221. score = 0;
222. step = 5;
223. GLCD\_Bargraph(eda.x, eda.y, step, step, Purple);
224. }
226. void vector\_set(unsigned int dj)
227. {
228. if ((dj & (1<<23)) == 0)
229. {
230. **if (vector != Down)**
231. vector = Up;
232. }
233. if ((dj & (1<<24)) == 0)
234. {
235. **if (vector != Left)**
236. vector = Right;
237. }
238. if ((dj & (1<<25)) == 0)
239. {
240. **if (vector != Up)**
241. vector = Down;
242. }
243. if ((dj & (1<<26)) == 0)
244. {
245. **if (vector != Right)**
246. vector = Left;
247. }
248. }
250. **void game\_over()**
251. {
252. GLCD\_DisplayString (4, 5, "GAME OVER");
253. while(1){
254. djostic = LPC\_GPIO1->FIOPIN & (1<<20);
255. **if(djostic == 0)**
256. {
257. break;
258. }
259. }
260. **display\_line();**
261. coord\_init();
262. }
264. void init\_ShIM()
265. **{**
266. LPC\_PINCON->PINSEL4 |= (0x01<<4) ; // P2.2 - out PWM1.3
267. //LPC\_GPIO2->FIOSET |= (1<<3) ;
269. LPC\_PWM1->PCR |= 0x01<<11 ; //Enable PWM1 3 output
270. **LPC\_PWM1->MCR = 0x00000002 ; //On match with timer reset the counter**
271. LPC\_PWM1->PR = 0x00000000 ; //Load prescaler
272. LPC\_PWM1->MR0 = 16000 ; //set cycle rate to one second
273. LPC\_PWM1->MR3 = 2000; //set duty cycle to 0%
274. LPC\_PWM1->LER = (1<<3)|(1<<0) ; //enable shadow latch for match 0
275. **LPC\_PWM1->TCR = 0x00000002 ; //Reset counter and prescaler**
276. LPC\_PWM1->TCR = 0x00000009 ; //enable counter and PWM, release counter from reset
277. }

# Вывод

В данной работе мы реализовали игру змейка и научились работать с АЦП.