```
1   //Kunal Mukherjee
2   //10/9/10
3   #include <at89c51cc03.h>
4   #include <stdio.h>
5   #include <stdlib.h>
6
7   //*Global functions
8
9   //*Global functions for hardware code
10  void SetLed(unsigned char row,
11              unsigned char column,
12              unsigned char state);
13  unsigned char GetLedStatus(unsigned char row,
14                             unsigned char column);
15  void DoLED(unsigned char r, unsigned char c);
16  unsigned char getUpDn (void);
17  void makeInitialGenUp(void);
18  void makeInitialGenDown(void);
19  unsigned char checkStatus(unsigned char row,
20                            unsigned char col);
21  void OutputEncoder(unsigned char rfRow);
22  void makeHrGlass(void);
23  //external assembly lanaguge program
24  extern int Assem(); //return 0
25
26  //*initialize global constants
27  unsigned char down = 0;
28  unsigned char up = 1;
29  unsigned char flat = 2;
30
31  //*initialize global variables
32  unsigned char LED[16];
33  unsigned char refreshRow = 0; //row to be refreshed
34
35  int main()
36  {
37    //*Main code
38
39    //*local variable
40    int i, ledRow, ledCol;
41
42    //*Initialize timer 0 interrupt
43    //set the clock control register to double clock
44    CKCON = 0x01;
45    //set up Timer 0 in the 16-bit auto-reload mode,
46    //not gated, with an internal clock
47    TMOD = 0x01;
48    //calculate value for timer zero
49    //refresh rate 2 ms = 2000 us
50    //2000/.2127 = 9402 counts
51    //65536-9402 = 56134 counts
52    //56134 = 0xDB46
53    TH0 = 0xDB;
54    TL0 = 0x46;
55
56    //*Set up timer 0 for multplexing
57    //enable T0 interrupt and the global interrupt
58    TR0 = 1;
59    ET0 = 1;
60    EA =1;
61
62    //*Set up A/D converter for Tilt sensor
63    ADCF  = 0x01 ; //P1.0 = ADC[0]
64    ADCON = 0X20; //enable ADC Function
65    ADCLK = 0X00; //Prescalar to 0
66
67    //*Initialize varibales
68
69    //*Clear LED memeory map
70     for (i = 0; i <16; i++)
71    {
72      LED[i] = 0x00;
```

```
73      }
74
75      //Place hour glass outlinne in memory map
76      if (getUpDn() == down)
77           makeInitialGenUp();
78       else
79           makeInitialGenDown();
80
81      //*Place hour glass outlinne in memory map
82      makeHrGlass();
83
84       //*Start timer for multiplexing display
85      //turn on timer 0
86      TR0 = 1;
87
88      //*main program loop
89      while(1) //user != 100
90        {
91          if(getUpDn() == down) //Start at top and move down
92          {
93              for (ledRow = 15; ledRow >= Assem(); ledRow--)
94            {
95              for (ledCol = 7; ledCol >= Assem(); ledCol--)
96              {
97                DoLED(ledRow,ledCol);
98              }
99            }
100         }
101         else //STart at bottom and move up
102         {
103           for (ledRow = Assem(); ledRow < 16; ledRow++)
104           {
105             for (ledCol = Assem(); ledCol < 8; ledCol++)
106             {
107               DoLED(ledRow, ledCol);
108             }
109           }
110         }
111
112         while(P1_5 == Assem()); //if stopped wait here user, pause  feature
113                                 //P1_5 = 0
114
115         if(P1_4 == Assem()) //P1_4 = 0 //reset feature
116           {
117             //*Clear LED memeory map
118                for (i = Assem(); i <16; i++)
119               {
120                 LED[i] = 0x00;
121               }
122
123             //*Place hour glass outlinne in memory map
124               if (getUpDn() == down)
125               {
126                 makeInitialGenUp();
127               }
128               else if(getUpDn() == up)
129               {
130                 makeInitialGenDown();
131               }
132
133               makeHrGlass();
134           }
135       }
136
137   }
138
139   void DoLED(unsigned char r, unsigned char c)
140     {
141
142       //printf("r: %i c: %i status:%i\n", r,c, GetLedStatus(r,c));
143       if(getUpDn() != flat)
144       {
```

```
145         if (getUpDn()== down)
146         {
147           //look at the bottom LED
148           //Move LED at (r,c) accordign to algirithm
149           //if not fill, move led
150           if((GetLedStatus(r,c) != 0) &&
151                       (r+1 < 16) &&
152                       (GetLedStatus(r+1,c) == 0) &&
153                       (checkStatus(r+1,c) != 0))
154           {
155             SetLed(r+1, c, 1);
156             SetLed(r  , c, 0);
157           }
158           else
159           {
160             //look at bottom left
161             //if not fill, move led
162             if((GetLedStatus(r,c) != 0) &&
163                           (r+1 < 16) &&
164                           (c-1 >= 0) &&
165                           (GetLedStatus(r+1,c - 1) == 0) &&
166                           (checkStatus(r+1,c - 1) != 0))
167             {
168               SetLed(r+1, c - 1, 1);
169               SetLed(r  , c, 0);
170             }
171           //look at bottom right
172           //if not fill, move led
173             else if((GetLedStatus(r,c) != 0) &&
174                             (r+1 < 16)&&
175                             (c+1 <= 7)&&
176                             (GetLedStatus(r+1,c + 1) == 0) &&
177                             (checkStatus(r+1,c + 1) != 0))
178             {
179               SetLed(r+1, c + 1, 1);
180               SetLed(r  , c, 0);
181             }
182           }
183         }
184         else
185         {
186           //look at the top LED
187           //Move LED at (r,c) accordign to algirithm
188           //if not fill, move led
189           if((GetLedStatus(r,c) != 0) &&
190                       (r-1 >= 0) &&
191                       (GetLedStatus(r-1,c) == 0) &&
192                       (checkStatus(r-1,c) != 0))
193           {
194             SetLed(r-1, c, 1);
195             SetLed(r  , c, 0);
196           }
197           else
198           {
199             //look at bottom left
200             //if not fill, move led
201             if((GetLedStatus(r,c) != 0) &&
202                           (r-1 >= 0) &&
203                           (c-1 >= 0) &&
204                           (GetLedStatus(r-1,c - 1) == 0) &&
205                           (checkStatus(r-1,c - 1) != 0))
206             {
207               SetLed(r-1, c - 1, 1);
208               SetLed(r  , c, 0);
209             }
210           //look at bottom right
211           //if not fill, move led
212             else if((GetLedStatus(r,c) != 0) &&
213                             (r-1 >= 0) &&
214                             (c+1 <= 7) &&
215                             (GetLedStatus(r-1,c + 1) == 0) &&
216                             (checkStatus(r-1,c+1) != 0))
```

```
217             {
218               SetLed(r-1, c + 1, 1);
219               SetLed(r  , c, 0);
220             }
221           }
222         }
223       }
224     }
225
226   //gets the led status of the row and col
227   unsigned char GetLedStatus(unsigned char row,
228                              unsigned char column)
229   {
230     unsigned char temp = LED[row];
231
232     return temp & (1 << column);
233   }
234
235   //sets the led at the point
236   void SetLed(unsigned char row,
237               unsigned char column,
238               unsigned char state)
239   {
240     if(state)
241       LED[row] |= (1 << column);
242     else
243       LED[row] &= ~(1 << column);
244   }
245
246   //looks at the accelerometer and gives a result out
247   unsigned char getUpDn (void)
248   {
249       unsigned char tmp;
250       int i,result;
251
252       ADCON &= 0xF8; // Reset ADC Channel Select
253       ADCON |= 0x00; // Select ADC = Ch0
254       ADCON |= 0x20; // Use Standard mode
255       ADCON |= 0x08; // Start ADC Convert
256
257       tmp = (ADCON & 0x10); // Get done bit
258       while(tmp != 0x10) // Loop until complete
259         tmp = (ADCON & 0x10);
260       result = ADDH; // Send 8 MSB to P2
261       result *= 4;
262       result += ADDL;
263
264       ADCON &= 0xEF; //clear ADEOC = 0
265
266       for (i = 0; i < 33; i++);
267
268       //return down;
269       if (result > 535)
270         return down;
271       else if (result < 380)
272         return up;
273       else
274         return flat;
275   }
276
277   //generates the hour glass
278   void makeHrGlass(void)
279     {
280
281       SetLed(5,7, 0);
282       SetLed(6,7, 0);
283       SetLed(7,7, 0);
284       SetLed(8,7, 0);
285       SetLed(9,7, 0);
286       SetLed(10,7, 0);
287
288       SetLed(5,0, 0);
```

```
289       SetLed(6,0, 0);
290       SetLed(7,0, 0);
291       SetLed(8,0, 0);
292       SetLed(9,0, 0);
293       SetLed(10,0, 0);
294
295       SetLed(6,6, 0);
296       SetLed(7,6, 0);
297       SetLed(8,6, 0);
298       SetLed(9,6, 0);
299
300       SetLed(6,1, 0);
301       SetLed(7,1, 0);
302       SetLed(8,1, 0);
303       SetLed(9,1, 0);
304
305       SetLed(7,5, 0);
306       SetLed(8,5, 0);
307
308       SetLed(7,2, 0);
309       SetLed(8,2, 0);
310     }
311
312       ///check to see if the location to go, is
313     //part of the hour glass or not
314   unsigned char checkStatus(unsigned char row,
315                             unsigned char col)
316   {
317      if ((row == 5) || (row == 6) || (row == 7) ||
318          (row == 8) || (row == 9) || (row == 10))
319      {
320          if((col == 0) || (col == 7))
321          {
322              return 0;
323          }
324      }
325
326      if  ((row == 6) || (row == 7) ||
327           (row == 8) || (row == 9))
328       {
329           if((col == 6) || (col == 1))
330          {
331              return 0;
332          }
333       }
334
335       if((row == 7) ||
336           (row == 8))
337       {
338           if((col == 5) || (col == 2))
339          {
340              return 0;
341          }
342       }
343
344       return 1;
345
346   }
347
348   //Makes the initial generation for Up
349   void makeInitialGenUp(void)
350   {
351       int i , j;
352       for (i = 0; i < 8; i++)
353       {
354           for (j = 0; j < 7; j++)
355           {
356               SetLed(j,i,1);
357           }
358       }
359   }
360
```

```
361   //Makes the initial generation for Down
362   void makeInitialGenDown(void)
363   {
364       int i , j;
365       for (i = 0; i < 8; i++)
366       {
367           for (j = 15; j > 8; j--)
368           {
369               SetLed(j,i,1);
370           }
371       }
372   }
373
374
375     //*Timer 0 multiplexor interrupt
376     //Convert refreshRow to port bits for decoders
377     //Make decoders active
378     //Send d to the 74LS244 port
379     void T0Int() interrupt 1 using 1
380     {
381       unsigned char d;
382       //*reload the timer value
383       //calculate value for timer zero
384       //refresh rate 2 ms = 2000 us
385       //2000/.2127 = 9402 counts
386       //65536-9402 = 56134 counts
387       //56134 = 0xDB46
388       TH0 = 0xDB;
389       TL0 = 0x46;
390
391       //* Get the data d = LED[refreshRow]
392       d = LED[refreshRow];
393       //Send d to the 74LS244 port
394       P3 = d;
395
396       //convert refreshRow to port bits for decoder
397       OutputEncoder(refreshRow);
398
399       //Update refreshRow
400       refreshRow++;
401       if(refreshRow == 16)
402         refreshRow = 0;
403     }
404
405     // Assume we have two 3x8 decoders, 74LS138
406     //both of which are connected to P1.4, P1.5, P1.7,
407     //with their enable pins being (P0.1, and P0.2)
408
409     void OutputEncoder(unsigned char rfRow)
410     {
411       // Handle choosing the right encoder based on MSB
412       P0_2 = P0_3 = 0;
413       if((rfRow & (1 << 3)) == 0)
414         P0_3 = 1;
415       else
416         P0_2  = 1;
417
418       // Output the least significan 3 bits from
419       //the input number onto the encoders input
420       P0_7 = ((rfRow >> 2) & 1);
421       P0_6 = ((rfRow >> 1) & 1);
422       P0_5 = ((rfRow >> 0) & 1);
423
424     }
425
426
```

```
1   public _Assem
2       ;Define a relocatable code segment named Assemb
3       Assemb SEGMENT CODE
4           ;Select Assemb as the active segment
5           RSEG Assemb
6               ;entry point of the C langauge call
7           _Assem: mov R6, #0; ;return 0 as int
8                   mov R7, #0;
9           ret
10          end
11
```