```c
//Kunal Mukherjee
//Project 2
//11/24/18
#include "stm32f446.h"
#include <stdio.h>
#include <stdlib.h>

/*stm446Template.c July 1, 2017
*/

//*Global functions

//*Global functions for hardware code
void SetLed(unsigned char row, unsigned char column, unsigned char state);
unsigned char GetLedStatus(unsigned char row, unsigned char column);
void DoLED1(unsigned char r, unsigned char c);
void DoLED2(unsigned char r, unsigned char c);
void DoLED3(unsigned char r, unsigned char c);
unsigned char getUpDn (void);
void makeInitialGenUp(void);
void makeInitialGenDown(void);
void makeInitialGenUpWater(void);
void makeInitialGenDownWater(void);
unsigned char checkStatus(unsigned char row, unsigned char col);
unsigned char checkStatus1(unsigned char row,unsigned char col);
int getPachinokoComplete(void);
void makeHrGlass(void);
void makeWaterGlass(void);
void OutputEncoder(unsigned char rfRow);

extern void checkStatus(void);


//*initialize global constants
unsigned char down = 0;
unsigned char up = 1;
unsigned char flat = 2;
int pachinkoRoundflag = 1;

unsigned char LED[16];
unsigned char refreshRow = 0; //row to be refreshed
int flag = 0;
int flag1 = 0;
int flag2 = 0;
int flag3 = 0;

int main()
 {
  //temporary variables initialization
  int i,it,jt, ledRow, ledCol, randCol;

  //Enabling Clock bits
  RCC_AHB1ENR |= 1;     //Bit 0 is GPIOA clock enable bit
  RCC_AHB1ENR |= 2; //Bit 1 is GPIOB clock enable bit
  RCC_APB1ENR |= 1; //Enable peripheral timer for timer 2 (bit 0)
  RCC_AHB1ENR |= 4;     //Bit 3 is GPIOC clock enable bit

  //I/O bits
  GPIOA_MODER |= 0x5C00;  //Bits 0-1 for Input,
  GPIOC_MODER |= 0x5555;    //Bits 0-7 for digitial output
  GPIOB_MODER |= 0x15;


  //OTYPER register resets to 0 so it is push/pull by default
  GPIOA_OSPEEDER |= 0xFFFFFF;   //Bits 0-15 for high speed on PA 1
  GPIOC_OSPEEDER |= 0xFFFF;     //Bits 0-7 for high speed on PA 3
  GPIOB_OSPEEDER |= 0x3003F;

  //Accelerometer setup
  RCC_APB2ENR |= 0x100;      //Bit 8 is ADC 1 clock enable bit
  GPIOA_MODER |= 0xC00;      //PA5 are analog
  GPIOA_PUPDR &= 0xFFFFF3FF; //Pins P5 are no pull up and pull down
```

```
73
74      ADC1_CR2 |= 1;              //Bit 0 turn ADC on
75      ADC1_CR2 |= 0x400;          //Bit 10 allows EOC to be set after conversion
76      ADC_CCR |= 0x30000;         //Bits 16 and 17 = 11 so clock divided by 8
77      ADC1_SQR3 |= 0x5;           //Bits 4:0 are channel number for first conversion
78      // Channel is set to 5 which corresponds to PA5
79
80
81      //Interrupt bits
82      NVICISER0 |= (1 << 28); //Bit 28 in ISER0 corresponds to int 28 (TIM 2)
83      TIM2_DIER |= 1; //Enable Timer 2 update interrupt enable
84      TIM2_DIER |= (1 << 6); //Enable Timer 2 trigger interrupt enable
85
86      //Timer 2 bits
87      TIM2_CR1 |= (1 << 7); //Auto reload is buffered
88      TIM2_PSC = 0; //Don't use prescaling
89      TIM2_ARR = 32000; //16MHz/32000 = 500 Hz
90      TIM2_CR1 |= 1; //Enable Timer 2
91      TIM2_EGR |= 1;
92
93      //*Initialize varibales
94
95      //*Clear LED memeory map
96       for (i = 0; i <16; i++)
97      {
98        LED[i] = 0;
99      }
100
101      //Main program loop
102      while(1)
103      {
104        //if interrupt triggered then turn flag off and restart timer
105       if (flag)
106       {
107         TIM2_CR1 |= 1; //Restart timer
108         flag = 0;
109       }
110
111       //if A0 == 1 then Algorithm 1
112       if (GPIOA_IDR & (1 << 0))
113       {
114         if(flag1 == 0)
115         {
116           flag1 = 1;
117           flag2 = 0;
118           flag3 = 0;
119           pachinkoRoundflag = 1;
120
121           //*Clear LED memeory map
122           for (i = 0; i <16; i++)
123           {
124             LED[i] = 0x00;
125           }
126
127          //*Place hour glass outlinne in memory map
128           if (getUpDn() == down)
129             makeInitialGenUp();
130           else
131             makeInitialGenDown();
132
133           makeHrGlass();
134
135         }
136
137         if(getUpDn() == down) //Start at top and move down
138               {
139                   for (ledRow = 15; ledRow >= 0; ledRow--)
140                   {
141                       for (ledCol = 0; ledCol <8; ledCol++)
142                       {
143                           DoLED1(ledRow,ledCol);
144                       }
```

```
145                 }
146             }
147             else //STart at bottom and move up
148             {
149                 for (ledRow = 0; ledRow < 16; ledRow++)
150                 {
151                     for (ledCol = 0; ledCol < 8; ledCol++)
152                     {
153                         DoLED1(ledRow, ledCol);
154                     }
155                 }
156             }
157 
158             for (it = 0; it < 64000; it++)
159             {
160               for (jt = 0; jt < 5; jt++);
161             }
162     }
163 
164 
165 
166     //if A1 == 1 then Algorithm 2
167     else if (GPIOA_IDR & (1 << 1))
168     {
169       if(flag2 == 0 || getPachinokoComplete())
170       {
171         flag2 = 1;
172         flag1 = 0;
173         flag3 = 0;
174         pachinkoRoundflag = 1;
175         //*Clear LED memeory map
176         for (i = 0; i <16; i++)
177         {
178           LED[i] = 0x00;
179         }
180       }
181 
182       if(getUpDn() == down) //Start at top and move down
183                 {
184                     randCol = (int) rand() % 8;
185                     if(pachinkoRoundflag == 1)
186                     {
187                         SetLed(0, randCol, 1);
188                         pachinkoRoundflag = 0;
189                     }
190 
191                     for (ledRow = 15; ledRow >= 0; ledRow--)
192                     {
193                         for (ledCol = 0; ledCol <8; ledCol++)
194                         {
195                             DoLED2(ledRow,ledCol);
196                         }
197                     }
198                 }
199                 else //STart at bottom and move up
200                 {
201                     randCol = (int) rand() % 8;
202                     if(pachinkoRoundflag == 1)
203                     {
204                         SetLed(15, randCol, 1);
205                         pachinkoRoundflag = 0;
206                     }
207 
208                     for (ledRow = 0; ledRow < 16; ledRow++)
209                     {
210                         for (ledCol = 0; ledCol < 8; ledCol++)
211                         {
212                             DoLED2(ledRow, ledCol);
213                         }
214                     }
215                 }
216 
```

```
217               for (it = 0; it < 64000; it++)
218               {
219                 for (jt = 0; jt < 2; jt++);
220               }
221
222       }
223
224
225
226       //if B8 == 1 then Algorithm 3
227       else if (GPIOB_IDR & (1 << 8))
228       {
229
230         if(flag3 == 0)
231         {
232           flag3 = 1;
233           flag2 = 0;
234           flag1 = 0;
235           pachinkoRoundflag = 1;
236           //*Clear LED memeory map
237           for (i = 0; i <16; i++)
238           {
239             LED[i] = 0x00;
240           }
241
242          //*Place hour glass outlinne in memory map
243           if (getUpDn() == down)
244           {
245             makeInitialGenUpWater();
246           }
247           else
248           {
249             makeInitialGenDownWater();
250           }
251
252           makeWaterGlass();
253
254         }
255
256
257         if(getUpDn() == down) //Start at top and move down
258               {
259                   for (ledRow = 15; ledRow >= 0; ledRow--)
260                   {
261                       for (ledCol = 0; ledCol <8; ledCol++)
262                       {
263                           DoLED3(ledRow,ledCol);
264                       }
265                   }
266               }
267               else //STart at bottom and move up
268               {
269                   for (ledRow = 0; ledRow < 16; ledRow++)
270                   {
271                       for (ledCol = 0; ledCol < 8; ledCol++)
272                       {
273                           DoLED3(ledRow, ledCol);
274                       }
275                   }
276               }
277
278          for (it = 0; it < 64000; it++)
279               {
280                 for (jt = 0; jt < 5; jt++);
281               }
282       }
283
284     }
285
286   }
287
288
```

```
289    //timer intreupt handler
290   void TIM2_IRQHandler()
291    {
292      unsigned char d;
293
294     GPIOC_ODR &= ~(1 << 0); //C0 = 0
295     GPIOC_ODR &= ~(1 << 1); //C1 = 0
296     GPIOC_ODR &= ~(1 << 2); //C2 = 0
297     GPIOC_ODR &= ~(1 << 3); //C3 = 0
298     GPIOC_ODR &= ~(1 << 4); //C4 = 0
299     GPIOC_ODR &= ~(1 << 5); //C5 = 0
300     GPIOC_ODR &= ~(1 << 6); //C6 = 0
301     GPIOC_ODR &= ~(1 << 7); //C7 = 0
302
303     //* Get the data d = LED[refreshRow]
304     d = LED[refreshRow];
305     GPIOC_ODR |= d;
306
307     //convert refreshRow to port bits for decoder
308     OutputEncoder(refreshRow);
309
310     //Update refreshRow
311     refreshRow++;
312     if(refreshRow == 16)
313       refreshRow = 0;
314
315     flag = 1;
316     TIM2_SR &= 0xFFFE; //Turn off interrupt
317
318    }
319
320
321   //the output encoder algorithms for 74LS244
322
323   void OutputEncoder(unsigned char rfRow)
324    {
325      // Handle choosing the right encoder based on MSB
326      GPIOA_ODR &= ~(1 << 6); //A6 = 0
327      GPIOA_ODR &= ~(1 << 7); //A7 = 0
328
329
330      if((rfRow & (1 << 3)) == 0)
331        GPIOA_ODR |= (1 << 6); //A6 = 1
332      else
333        GPIOA_ODR |= (1 << 7); //A7 = 1
334
335      // Output the least significan 3 bits from
336      //the input number onto the encoders input
337      //CLEAR C0,C1,2
338      GPIOB_ODR &= ~(1 << 0); //A8 = 0
339      GPIOB_ODR &= ~(1 << 1); //A9 = 0
340      GPIOB_ODR &= ~(1 << 2); //A10 = 0
341
342
343      GPIOB_ODR |= (((rfRow >> 2) & 1) << 2);//((((rfRow >> 0) & 1) << 0);
344      GPIOB_ODR |= (((rfRow >> 1) & 1) << 1);//((((rfRow >> 1) & 1) << 1);
345      GPIOB_ODR |= (((rfRow >> 0) & 1) << 0);//((((rfRow >> 2) & 1) << 2);
346    }
347
348   unsigned char getUpDn (void)
349  {
350    int result;
351    ADC1_CR2 |= 0x40000000;      // Bit 30 does software start of A/D conversion
352    while((ADC1_SR & 0x2) == 0); //Bit 1 is End of Conversion
353
354    result = ADC1_DR & 0xFFF;
355
356    if  (result > 2023)
357      return up;
358    else if(result < 1737)
359      return down;
360    else
```

```
361       return flat;
362
363      //return down;
364   }
365
366
367   //hourglass algorithm
368    void DoLED1(unsigned char r, unsigned char c)
369     {
370
371       //printf("r: %i c: %i status:%i\n", r,c, GetLedStatus(r,c));
372       if(getUpDn() != flat)
373       {
374         if (getUpDn()== down)
375         {
376           //look at the bottom LED
377           //Move LED at (r,c) accordign to algirithm
378           //if not fill, move led
379           if((GetLedStatus(r,c) != 0) &&
380                       (r+1 < 16) &&
381                       (GetLedStatus(r+1,c) == 0) &&
382                       (checkStatus(r+1,c) != 0))
383           {
384             SetLed(r+1, c, 1);
385             SetLed(r  , c, 0);
386           }
387           else
388           {
389             //look at bottom left
390             //if not fill, move led
391             if((GetLedStatus(r,c) != 0) &&
392                           (r+1 < 16) &&
393                           (c-1 >= 0) &&
394                           (GetLedStatus(r+1,c - 1) == 0) &&
395                           (checkStatus(r+1,c - 1) != 0))
396             {
397               SetLed(r+1, c - 1, 1);
398               SetLed(r  , c, 0);
399             }
400           //look at bottom right
401           //if not fill, move led
402             else if((GetLedStatus(r,c) != 0) &&
403                             (r+1 < 16)&&
404                             (c+1 <= 7)&&
405                             (GetLedStatus(r+1,c + 1) == 0) &&
406                             (checkStatus(r+1,c + 1) != 0))
407             {
408               SetLed(r+1, c + 1, 1);
409               SetLed(r  , c, 0);
410             }
411           }
412         }
413         else
414         {
415           //look at the top LED
416           //Move LED at (r,c) accordign to algirithm
417           //if not fill, move led
418           if((GetLedStatus(r,c) != 0) &&
419                       (r-1 >= 0) &&
420                       (GetLedStatus(r-1,c) == 0) &&
421                       (checkStatus(r-1,c) != 0))
422           {
423             SetLed(r-1, c, 1);
424             SetLed(r  , c, 0);
425           }
426           else
427           {
428             //look at bottom left
429             //if not fill, move led
430             if((GetLedStatus(r,c) != 0) &&
431                           (r-1 >= 0) &&
432                           (c > 0) &&
```

```
433                           (GetLedStatus(r-1,c - 1) == 0) &&
434                           (checkStatus(r-1,c - 1) != 0))
435             {
436               SetLed(r-1, c - 1, 1);
437               SetLed(r  , c, 0);
438             }
439           //look at bottom right
440           //if not fill, move led
441             else if((GetLedStatus(r,c) != 0) &&
442                               (r-1 >= 0) &&
443                               (c < 7) &&
444                               (GetLedStatus(r-1,c + 1) == 0) &&
445                               (checkStatus(r-1,c+1) != 0))
446             {
447               SetLed(r-1, c + 1, 1);
448               SetLed(r  , c, 0);
449             }
450           }
451         }
452       }
453     }
454
455
456     //Pachinko algorithm
457   void DoLED2(unsigned char r, unsigned char c)
458     {
459         if(getUpDn() != flat)
460       {
461         if (getUpDn()== down)
462         {
463           //look at the bottom LED
464           //Move LED at (r,c) accordign to algirithm
465           //if not fill, move led
466           if((GetLedStatus(r,c) != 0) &&
467                       (r+1 < 16) &&
468                       (GetLedStatus(r+1,c) == 0))
469           {
470             SetLed(r+1, c, 1);
471             SetLed(r  , c, 0);
472             //to check if bottom is hit
473             if (r+1 == 15 || GetLedStatus(r+2,c) != 0)
474                           pachinkoRoundflag = 1;
475           }
476           else
477           {
478             //look at bottom left
479             //if not fill, move led
480             if((GetLedStatus(r,c) != 0) &&
481                           (r+1 < 16) &&
482                           (c-1 >= 0) &&
483                           (GetLedStatus(r+1,c - 1) == 0))
484             {
485               SetLed(r+1, c - 1, 1);
486               SetLed(r  , c, 0);
487
488               if (r+1 == 15 || GetLedStatus(r+2,c-1) != 0)
489                             pachinkoRoundflag = 1;
490             }
491           //look at bottom right
492           //if not fill, move led
493             else if((GetLedStatus(r,c) != 0) &&
494                               (r+1 < 16)&&
495                               (c+1 <= 7)&&
496                               (GetLedStatus(r+1,c + 1) == 0))
497             {
498               SetLed(r+1, c + 1, 1);
499               SetLed(r  , c, 0);
500
501               if (r+1 == 15 || GetLedStatus(r+2,c+1) != 0)
502                             pachinkoRoundflag = 1;
503             }
504           }
```

```
505         }
506         else
507         {
508           //look at the top LED
509           //Move LED at (r,c) according to algorithm
510           //if not fill, move led
511           if((GetLedStatus(r,c) != 0) &&
512                       (r-1 >= 0) &&
513                       (GetLedStatus(r-1,c) == 0))
514           {
515
516             SetLed(r-1, c, 1);
517             SetLed(r  , c, 0);
518
519             if (r-1 == 0 || GetLedStatus(r-2,c) != 0)
520                       {
521                           pachinkoRoundflag = 1;
522                       }
523
524           }
525           else
526           {
527             //look at bottom left
528             //if not fill, move led
529             if((GetLedStatus(r,c) != 0) &&
530                           (r-1 >= 0) &&
531                           (c > 0) &&
532                           (GetLedStatus(r-1,c - 1) == 0))
533             {
534
535               SetLed(r-1, c - 1, 1);
536               SetLed(r  , c, 0);
537
538               if (r-1 == 0 || GetLedStatus(r-2,c-1) != 0)
539                           {
540                               pachinkoRoundflag = 1;
541                           }
542
543             }
544           //look at bottom right
545           //if not fill, move led
546             else if((GetLedStatus(r,c) != 0) &&
547                               (r-1 >= 0) &&
548                               (c < 7) &&
549                               (GetLedStatus(r-1,c + 1) == 0))
550             {
551
552               SetLed(r-1, c + 1, 1);
553               SetLed(r  , c, 0);
554
555               if (r-1 == 0 || GetLedStatus(r-2,c + 1) != 0)
556                           {
557                               pachinkoRoundflag = 1;
558                           }
559             }
560           }
561         }
562       }
563     }
564
565
566     //Waterfall algorithm
567   void DoLED3(unsigned char r, unsigned char c)
568     {
569
570       //printf("r: %i c: %i status:%i\n", r,c, GetLedStatus(r,c));
571       if(getUpDn() != flat)
572       {
573         if (getUpDn()== down)
574         {
575           //look at the bottom LED
576           //Move LED at (r,c) accordign to algirithm
```

```
577           //if not fill, move led
578           if((GetLedStatus(r,c) != 0) &&
579                       (r+1 < 16) &&
580                       (GetLedStatus(r+1,c) == 0) &&
581                       (checkStatus1(r+1,c) != 0))
582           {
583             SetLed(r+1, c, 1);
584             SetLed(r  , c, 0);
585           }
586           else
587           {
588             //look at bottom left
589             //if not fill, move led
590             if((GetLedStatus(r,c) != 0) &&
591                           (r+1 < 16) &&
592                           (c-1 >= 0) &&
593                           (GetLedStatus(r+1,c - 1) == 0) &&
594                           (checkStatus1(r+1,c - 1) != 0))
595             {
596               SetLed(r+1, c - 1, 1);
597               SetLed(r  , c, 0);
598             }
599           //look at bottom right
600           //if not fill, move led
601             else if((GetLedStatus(r,c) != 0) &&
602                               (r+1 < 16)&&
603                               (c+1 <= 7)&&
604                               (GetLedStatus(r+1,c + 1) == 0) &&
605                               (checkStatus1(r+1,c + 1) != 0))
606             {
607               SetLed(r+1, c + 1, 1);
608               SetLed(r  , c, 0);
609             }
610
611             else if((GetLedStatus(r,c) != 0) &&
612                               (c+1 <= 7)&&
613                               (GetLedStatus(r,c + 1) == 0) &&
614                               (checkStatus1(r,c + 1) != 0))
615             {
616               SetLed(r, c + 1, 1);
617               SetLed(r  , c, 0);
618             }
619
620             else if((GetLedStatus(r,c) != 0) &&
621                               (c-1 >= 0)&&
622                               (GetLedStatus(r,c - 1) == 0) &&
623                               (checkStatus1(r,c - 1) != 0))
624             {
625               SetLed(r, c - 1, 1);
626               SetLed(r  , c, 0);
627             }
628           }
629         }
630         else
631         {
632           //look at the top LED
633           //Move LED at (r,c) accordign to algirithm
634           //if not fill, move led
635           if((GetLedStatus(r,c) != 0) &&
636                       (r-1 >= 0) &&
637                       (GetLedStatus(r-1,c) == 0) &&
638                       (checkStatus1(r-1,c) != 0))
639           {
640             SetLed(r-1, c, 1);
641             SetLed(r  , c, 0);
642           }
643           else
644           {
645             //look at bottom left
646             //if not fill, move led
647             if((GetLedStatus(r,c) != 0) &&
648                           (r-1 >= 0) &&
```

```
                          (c > 0) &&
                          (GetLedStatus(r-1,c - 1) == 0) &&
                          (checkStatus1(r-1,c - 1) != 0))
            {
              SetLed(r-1, c - 1, 1);
              SetLed(r  , c, 0);
            }
          //look at bottom right
          //if not fill, move led
            else if((GetLedStatus(r,c) != 0) &&
                                (r-1 >= 0) &&
                                (c < 7) &&
                                (GetLedStatus(r-1,c + 1) == 0) &&
                                (checkStatus1(r-1,c+1) != 0))
            {
              SetLed(r-1, c + 1, 1);
              SetLed(r  , c, 0);
            }

            else if((GetLedStatus(r,c) != 0) &&
                                (c < 7) &&
                                (GetLedStatus(r,c + 1) == 0) &&
                                (checkStatus1(r,c + 1) != 0))
            {
              SetLed(r, c + 1, 1);
              SetLed(r  , c, 0);
            }

            else if((GetLedStatus(r,c) != 0) &&
                                (c > 0) &&
                                (GetLedStatus(r,c - 1) == 0) &&
                                (checkStatus1(r,c - 1) != 0))
            {
              SetLed(r, c - 1, 1);
              SetLed(r  , c, 0);
            }
          }
        }
      }
    }


    //check to see if Panchino is Complete
  int getPachinokoComplete()
  {
      int ledRow, ledCol;
      for (ledRow = 2; ledRow < 14; ledRow++)
      {
          for (ledCol = 0; ledCol < 8; ledCol++)
          {
              if(!GetLedStatus(ledRow, ledCol))
                  return 0;
          }
      }
      return 1;
  }


  //check to see if the hour glass led status has hit or not
  unsigned char checkStatus(unsigned char row,
                            unsigned char col)
  {
     if ((row == 5) || (row == 6) || (row == 7) ||
         (row == 8) || (row == 9) || (row == 10))
     {
         if((col == 0) || (col == 7))
         {
             return 0;
         }
     }

```

```
721      if  ((row == 6) || (row == 7) ||
722           (row == 8) || (row == 9))
723       {
724           if((col == 6) || (col == 1))
725          {
726              return 0;
727          }
728       }
729
730       if((row == 7) ||
731           (row == 8))
732       {
733           if((col == 5) || (col == 2))
734          {
735              return 0;
736          }
737       }
738
739       return 1;
740
741   }
742
743
744   //check to see if it has hit a boundary
745   unsigned char checkStatus1(unsigned char row,
746                              unsigned char col)
747   {
748      if (row == 4)
749      {
750          if(    (col == 0) || (col == 1)
751             || (col == 2) || (col == 3)
752             || (col == 4) || (col == 5)
753             || (col == 6))
754          {
755              return 0;
756          }
757      }
758      if(row == 10)
759      {
760           if(  (col == 4) || (col == 5)
761             || (col == 6) || (col == 7)
762             || (col == 3) || (col == 2)
763             || (col == 1))
764          {
765              return 0;
766          }
767      }
768
769
770       return 1;
771
772   }
773
774
775   //get the led status of a led
776   unsigned char GetLedStatus(unsigned char row,
777                              unsigned char column)
778   {
779     unsigned char temp = LED[row];
780
781     return temp & (1 << column);
782   }
783
784
785   //set the led status
786   void SetLed(unsigned char row,
787               unsigned char column,
788               unsigned char state)
789   {
790     if(state)
791       LED[row] |= (1 << column);
792     else
```

```
793         LED[row] &= ~(1 << column);
794     }
795 
796 
797     //Make initial gen of hour glass up
798     void makeInitialGenUp(void)
799     {
800         int i , j;
801         for (i = 0; i < 8; i++)
802         {
803             for (j = 0; j < 7; j++)
804             {
805                 SetLed(j,i,1);
806             }
807         }
808     }
809 
810 
811     //Make initial gen of hour glass down
812     void makeInitialGenDown(void)
813     {
814         int i , j;
815         for (i = 0; i < 8; i++)
816         {
817             for (j = 15; j > 8; j--)
818             {
819                 SetLed(j,i,1);
820             }
821         }
822     }
823 
824 
825     //Make the initial generation of water up
826     void makeInitialGenUpWater(void)
827     {
828         int i , j;
829         for (i = 0; i < 4; i++)
830         {
831             for (j = 0; j < 7; j++)
832             {
833                 SetLed(i,j,1);
834             }
835         }
836     }
837 
838 
839     //Make the initial generation of water down
840     void makeInitialGenDownWater(void)
841     {
842         int i , j;
843         for (i = 11; i < 15; i++)
844         {
845             for (j = 1; j < 8; j++)
846             {
847                 SetLed(i,j,1);
848             }
849         }
850     }
851 
852 
853     //Make the Hour Glass outline
854     void makeHrGlass(void)
855       {
856 
857         SetLed(5,7, 0);
858         SetLed(6,7, 0);
859         SetLed(7,7, 0);
860         SetLed(8,7, 0);
861         SetLed(9,7, 0);
862         SetLed(10,7, 0);
863 
864         SetLed(5,0, 0);
```

```
865       SetLed(6,0, 0);
866       SetLed(7,0, 0);
867       SetLed(8,0, 0);
868       SetLed(9,0, 0);
869       SetLed(10,0, 0);
870 
871       SetLed(6,6, 0);
872       SetLed(7,6, 0);
873       SetLed(8,6, 0);
874       SetLed(9,6, 0);
875 
876       SetLed(6,1, 0);
877       SetLed(7,1, 0);
878       SetLed(8,1, 0);
879       SetLed(9,1, 0);
880 
881       SetLed(7,5, 0);
882       SetLed(8,5, 0);
883 
884       SetLed(7,2, 0);
885       SetLed(8,2, 0);
886     }
887 
888 
889     //Make the Water Glass Outline
890   void makeWaterGlass(void)
891   {
892       SetLed(4,0,0);
893       SetLed(4,1,0);
894       SetLed(4,2,0);
895       SetLed(4,3,0);
896       SetLed(4,4,0);
897       SetLed(4,5,0);
898       SetLed(4,6,0);
899 
900       SetLed(10,1,0);
901       SetLed(10,2,0);
902       SetLed(10,3,0);
903       SetLed(10,4,0);
904       SetLed(10,5,0);
905       SetLed(10,6,0);
906       SetLed(10,7,0);
907   }
908 
909 
910 
```