KOC UNIVERSITY

ELEC 317 / EMBEDDED SYSTEMS

SNAKE GAME

Deniz Karadayı 59925
Berke Başoğlu 60135

DATE:

05/01/2020

Introduction

Our project was designing a classic snake game that is controlled by a joystick, by using Assembly C. Our task was to use three attributes that we learned in the lecture. We used; SPI, Interrupt and ADC.

The equipment that we used in this project are:

- Atmega32
- MAX7219 8x8 R CLICK
- XY Joystick Module
- Female to Female Jumper Wires

Algorithm of the Code

We implemented the snake by using two different arrays. We have seven functions to run the snake. In the main function we only wrote the interrupts and the algorithm of the snake by using these sub-functions:

move (): This function makes the snake appear on the other end of the display when it comes to the end of the sides

moveSnake(): Moving the snake. All the bits of the snake follow the bit which is in front of itself.

updateDir(): Update the direction of the head of the snake when the user changes the direction.

IsSnakeDead(): Checking whether the snake is dead or not. We compared all bits' locations. If they were the same (overlapped) we assumed that the snake is dead.

IsFeedEaten(): Checking whether the snake ate the feed or not. We compared all bits' locations with the feed's locations. If they were the same (overlapped) we assumed that snake ate the feed.

snakeGrows(): If the snake ate the feed, we extended the length of the snake by 1 (we added the extended bit to the tail of the snake).

joyStickDirectChange(): We used ADC and interrupt in this function to check whether the user used the joystick or not. The code was very similar to the one on the last lab. The input voltage was between 0 and 5V. We considered that 2.5 and around 2.5 V corresponded to no movement.

We used one function to generate a feed:

random_dot(): We generated a random dot (feed) to the display by using a random generator sub-function which we have also implemented. Also, we checked whether the location of the feed and the snake overlapped or not. If it did, we generated another dot.

We used two functions to display a happy face that appears in the beginning of the game and a sad face that appears when the snake dies. Also, we used several basic sub-functions like "power", "absolute value" etc…

Problems Occurred:

In the beginning our main problem was how to use MAX7219 8x8 R CLICK. Atmega32 can receive max 8 bits of register where MAX7219 8x8 R CLICK sends 16 bits of register. Therefore, we couldn't ensure the communication between the MAX7219 8x8 R CLICK and Atmega32. We got help from Professor Engin Erzin and Mr. Selim. However, we couldn't solve those problems for several weeks. We couldn't find any libraries for Assembly. Finally, we found a library for Arduino and we arranged and changed the code of the library for Assembly and it didn't work. Then we noticed a register error in the library, and we fixed it. It worked after all.

Then we couldn't calibrate the joystick. We fixed the code several times but, it didn't work. We noticed that the wiring was wrong. We fixed it but again, it didn't work. We erased the entire joystick code and we wrote it from scratch. Finally, it worked.

We encountered with a lot of problems in this project. The project could have been finished two weeks earlier if we didn't encounter with any problems.

Conclusion

When we chose this project, we didn't think that it would be this hard. We even thought of changing the project. We said, "We will look at the code one last time and if we couldn't fix the problem, we will change the project" and we fix it. After that, we finally finished it and we think it improved us in many ways and we learned a lot of new things. It was challenging but it was also fun.

Also, we noticed that arrays are the worst attributes of both of us. This was bad coincidence which prolonged the time of the project and forced us.

In addition, we noticed that Assembly C is much easier than Assembly.

The Code

```
#define F_CPU 1000000UL // 1 MHz clock speed
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdlib.h>
#include <math.h>
#include <stdbool.h>
unsigned char dat,address,col,row,N,changeInDir;
char mean = 115;
unsigned char thold = 50;
int jsY,jsX;
unsigned char select=0;
unsigned char temp=0;
unsigned char snakeLength = 1;
unsigned char snakeLoc[20][2];
unsigned char dir[20];
unsigned char feed[2];
unsigned char LEDScreenBlocks[8];
char dirX,dirY;


void randomGenerator (){
	N = (N*5+7)%8;
}

void resetGame(){
	unsigned char i;
	snakeLength = 1;
	for(i=0; i<20; i++){
		dir[i]=0;
		if(i<8) LEDScreenBlocks[i]=0;
		snakeLoc[i][0]=0;
		snakeLoc[i][1]=0;
	}
	feed[1]=0;


}


int absulate(int c){
	if (c<0) return c*(-1);
	else return c;

}


int power(unsigned char x, unsigned char t){
	unsigned char i, tmp;
	tmp=1;
	for (i=0; i<t; i++){
		tmp=x*tmp;
	}
	return tmp;
```

```
}

void shiftOut(unsigned char reg)
{
	unsigned char i;

	for (i=0;i<8;i++)  {
		PORTB = ( (!! (reg & (1 << (7-i))))  <<  PB0 );
		PORTD |= (1 << PD4);
		PORTD &= ~(1 << PD4);
	}
}

void setCommand(unsigned char addr, unsigned char data){

	PORTD &= ~(1 << PD1);

	shiftOut(addr);
	shiftOut(data);

	PORTD &= ~(1 << PD1);
	PORTD |= (1 << PD1);
}
void clear(){
	for(int i=1; i<9;i++){
		setCommand(i,0);
	}

}
void clearArray(unsigned char array[8]){
	for(int i=0; i<8;i++){
		array[i]=0;
	}

}

void random_dot(){
	unsigned char i=0;
	randomGenerator();
	col=N;
	randomGenerator();
	row=N;

	for(i=0; i<snakeLength; i++){
		if(snakeLoc[i][0]==col && snakeLoc[i][1]==row)
			random_dot();
	}

	feed[0]=col;
	feed[1]=row;

}

/*
void button(){
	if(PINA == 0b10000000){
		dir[0]=2;
	}else if(PINA == 0b01000000){
			dir[0]=4;
		}else if(PINA == 0b00100000){
			dir[0]=3;
		}else if(PINA == 0b00010000){
			dir[0]=1;
		}

}*/
void printBye(){
		unsigned char i;
		for(i=0;i<3;i++){
			setCommand(1,0b00000001);
			setCommand(2,0b00000010);
			setCommand(3,0b01110100);
			setCommand(4,0b00000100);
			setCommand(5,0b00000100);
			setCommand(6,0b01110100);
			setCommand(7,0b00000010);
			setCommand(8,0b00000001);
			_delay_ms(300);
			setCommand(1,0b00000001);
			setCommand(2,0b00100010);
			setCommand(3,0b00010100);
			setCommand(4,0b00100100);
			setCommand(5,0b00100100);
			setCommand(6,0b00010100);
			setCommand(7,0b00100010);
			setCommand(8,0b00000001);
			_delay_ms(300);
		}
		setCommand(1,0b00000001);
		setCommand(2,0b00000010);
		setCommand(3,0b01110100);
		setCommand(4,0b00000100);
		setCommand(5,0b00000100);
		setCommand(6,0b01110100);
		setCommand(7,0b00000010);
		setCommand(8,0b00000001);
		_delay_ms(700);

}
void printHello(){
		unsigned char i;
		for(i=0;i<3;i++){
			setCommand(3,0b00111110);
			setCommand(5,0b00111110);
			setCommand(6,0b00001000);
			setCommand(7,0b00111110);
			_delay_ms(300);
		}
		_delay_ms(300);
		for(i=0;i<3;i++){
			setCommand(1,0b00001000);
			setCommand(2,0b00000100);
			setCommand(3,0b01110010);
			setCommand(4,0b00000010);
			setCommand(5,0b00000010);
			setCommand(6,0b01110010);
			setCommand(7,0b00000100);
			setCommand(8,0b00001000);
			_delay_ms(300);
			setCommand(1,0b00001000);
			setCommand(2,0b00100100);
			setCommand(3,0b00010010);
			setCommand(4,0b00100010);
			setCommand(5,0b00100010);
			setCommand(6,0b00010010);
			setCommand(7,0b00100100);
			setCommand(8,0b00001000);
			_delay_ms(300);
		}

		setCommand(1,0b00001000);
		setCommand(2,0b00000100);
		setCommand(3,0b01110010);
		setCommand(4,0b00000010);
		setCommand(5,0b00000010);
		setCommand(6,0b01110010);
		setCommand(7,0b00000100);
		setCommand(8,0b00001000);
		_delay_ms(700);

}


unsigned char joyStickDirectChange(){

		if(jsX>mean+thold){
			dirX=1;
		}else if(jsX<mean-thold){
			dirX=-1;
		}else{
			dirX=0;
		}

		if(jsY>mean+thold){
			dirY=1;
		}else if(jsY<mean-thold){
			dirY=-1;
		}else{
			dirY=0;
		}

		if (!(dirX==0 || dirY==0))
		{
			if(absulate (mean-jsX) > absulate (mean-jsY)){
				return (-1)*dirX + 2;
			}else{
				return dirY+3;
			}
		}
		else{
			if (dirX==0 && dirY==0)
			{
				return 0;
			}
```

```
              else if (dirX==0)
              {
                     return dirY+3;
              }
              else{
                     return (-1)*dirX+2;
              }
       }


       return 0;

}
void checkAndChangeDirection(){
       if(changeInDir!=0){
              dir[0]=changeInDir;
       }
}


// checkAndChangeDirection --> move --> updateDir (loop for not to confuse snake movement)

/*     1 --> RIGHT: 7 is right-most columnn (row(-1))
       3 --> LEFT: 0 is left-most columnn (row(+1))
       2 --> UP: 7 is up-most row (column(+1))
       4 --> DOWN: 0 is down-most row (column(-1))
                                                                                                 */
void move(int direc, int i){
       if(direc==1){
              // RIGHT
              if(snakeLoc[i][0]==0) snakeLoc[i][0]=7;
              else snakeLoc[i][0]--;

       }else if(direc==3){
              // LEFT
              if(snakeLoc[i][0]==7) snakeLoc[i][0]=0;
              else snakeLoc[i][0]++;

       }else if(direc==2){
              // UP
              if(snakeLoc[i][1]==7) snakeLoc[i][1]=0;
              else snakeLoc[i][1]++;

       }else{
              // DOWN
              if(snakeLoc[i][1]==0) snakeLoc[i][1]=7;
              else snakeLoc[i][1]--;

       }
}
void moveSnake(){

       for(int i=0; i<snakeLength; i++){
              move(dir[i],i);
       }
}
```

```
void updateDir(){    // change the directions for the next move, shifts the directions of
each point of snake.
       unsigned char tempDir[snakeLength];

       tempDir[0]=dir[0];

       for(int i=0; i<snakeLength-1; i++){
              tempDir[i+1]=dir[i];
       }
       for(int i=1; i<snakeLength; i++){
              dir[i]=tempDir[i];
       }
}

int IsSnakeDead(){
       if(snakeLength<5) return 0; // If length is lower than 5, snake cannot eat itself.
       unsigned char head[2]= { snakeLoc[0][0] , snakeLoc[0][1] };

       for(int i=4; i<snakeLength; i++){
              if(head[0]==snakeLoc[i][0]){
                     if(head[1]==snakeLoc[i][1]){
                            return 1;
                     }
              }
       }
       return 0;

}

int IsFeedEaten(){
       unsigned char head[2]= { snakeLoc[0][0] , snakeLoc[0][1] };

       if(dir[0]==1){
              if(head[0]==0){
                     if((feed[0]==7) && (feed[1]==head[1])) return 1;
                     else return 0;
       }else{
                     if((feed[0]==head[0]-1) && (feed[1]==head[1])) return 1;
                     else return 0;
       }

       }else if(dir[0]==3){
              if(head[0]==7){
                     if((feed[0]==0) && (feed[1]==head[1])) return 1;
                     else return 0;
              }else{
                     if((feed[0]==head[0]+1) && (feed[1]==head[1])) return 1;
                     else return 0;
              }

       }else if(dir[0]==2){
              if(head[1]==7){
                     if((feed[0]==head[0]) && (feed[1]==0)) return 1;
                     else return 0;
              }else{
                     if((feed[0]==head[0]) && (feed[1]==head[1]+1)) return 1;
                     else return 0;
              }
```

```
	}else{
		if(head[1]==0){
			if((feed[0]==head[0]) && (feed[1]==7)) return 1;
			else return 0;
		}else{
			if((feed[0]==head[0]) && (feed[1]==head[1]-1)) return 1;
			else return 0;
		}
	}
}

void snakeGrows(){
	snakeLength++;
	unsigned char tmpSnake[snakeLength][2];
	unsigned char i,j;
	tmpSnake[0][0]=feed[0];
	tmpSnake[0][1]=feed[1];
	for(i=0;i<snakeLength-1;i++){
		for(j=0;j<2;j++){
			tmpSnake[i+1][j]=snakeLoc[i][j];
		}
	}
	for(i=0;i<snakeLength;i++){
		for(j=0;j<2;j++){
			snakeLoc[i][j]=tmpSnake[i][j];
		}
	}
}
void coordinateToNumberBlocks(){

	unsigned char digit[8];
	clearArray(digit);

	for(int i=0; i<snakeLength; i++){
		col = snakeLoc[i][0];
		row = snakeLoc[i][1];
		digit[col] |=  power(2,row);
	}
	digit[feed[0]] |=  power(2,feed[1]);

	for(int i=0; i<8; i++){
		LEDScreenBlocks[i]=digit[i];
	}
}

void display(){
	coordinateToNumberBlocks();
	for(int i=0; i<8; i++){
		setCommand(i+1,LEDScreenBlocks[i]);
	}
}
void pause(){
	while(1){

	}
}
int main(void)
{
        //Initialize PORTS
        DDRB = 0xFF; // output
        DDRD = 0xFF; // output
        DDRC = 0xFF; // output
        DDRA = 0;      // input
        PORTD |= (1 << PD4);

        //Initialize MAXI7219
        setCommand(0x0b,0x07);
        setCommand(0x09,0x00);
        setCommand(0x0c,0x01);
        setCommand(0x0f,0x00);
        clear();

        // ADC Initialize
        ADCSRA= (1<<ADEN)|(1<<ADSC)|(1<<ADATE)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS0);
        ADMUX= (1<<REFS0)|(1<<ADLAR);
        sei();

        // Resetting
        resetGame();

        //Initialize Snake
        snakeLoc[0][0]=4;
        snakeLoc[0][1]=4;
        dir[0]=1;

        N=3;
        random_dot();
        printHello();



        while(1){
               display();
               _delay_ms(300);      // Interrupt ile x-y change bakıyor.

               checkAndChangeDirection();
               //button();
               if(IsFeedEaten()){
                      snakeGrows();
                      random_dot();
                      updateDir();
               }
               moveSnake();
               if(IsSnakeDead()){
                      display();
                      _delay_ms(200);
                      printBye();
                      main();
               }

               updateDir();

        }

        return 0;
}
```

```
ISR (ADC_vect) {
	if(select==0){
		temp=ADCL;
		jsX=ADCH;
		select=1;
		ADMUX = ADMUX | (1<<MUX1);
	}else{
		temp=ADCL;
		jsY=ADCH;
		select=0;
		ADMUX = ADMUX & ~(1<<MUX1);
	}

	changeInDir = joyStickDirectChange();
}
```