Framtidens spelkonsol

Ett projektarbete utfört av Pontus Stefansson och Oskar Ståhl

Sammanfattning

Vi har i kursen Digitala Projekt EITF11 byggt en spelkonsol på vilken man kan spela ”Snake”. Mjukvaran programmerades i C på en AVR Mega32 processor. Övrig hårdvara som användes var en 128x64-pixlars LCD-display, fem stycken knappar. Kraven på spelkonsolen var att spelaren ska kunna se sitt resultat samtidigt som spelet pågår, ormen ska växa under spelets gång och äpplen ska slumpmässigt placeras ut på skärmen. Projektets första fas gick ut på att montera och koppla samman all hårdvara. Detta gick relativt snabbt och utan större problem. När allting var på plats började arbetet med att programmera spelet. Det var i denna fas de största problemen påträffades. Så fort vi vant oss vid att programmera i C och hur man tar hand om signaler från processorn flöt emellertid arbetet på fint.

Innehåll

Inledning

Kursen Digitala Projekt ingår i I-programmets teknikprofil ”System och programvaruutveckling”. Målet med kursen är att ge studenterna en ökad inblick i hur konstruktions- och utvecklingsarbete genomförs. Detta uppnås genom ett projektarbete där målet är att ta fram en prototyp till en av gruppen vald produkt.

Vi i grupp 14 har valt att utveckla en konsol på vilken man kan spela det klassiska spelet Snake.

Kravspecifikation

Spelkonsolen skall uppfylla följande krav:

- Snake ska kunna spelas på konsolen, och spelet ska ha följande funktioner:
    - Ormen ska växa under spelets gång.
    - Äpplen ska generas och sedan placeras ut slumpmässigt på spelplanen.
    - Poängställningen ska presenteras i realtid på skärmen.
    - Då ormen åker in i någon av spelplanens väggar, eller i sig själv avbryts spelet.
    - Spelet ska gå att pausa.
    - Vid ”Game-Over” ska man kunna starta ett nytt spel.
- Spelet ska styras av en knappsats.
- Spelet ska presenteras på en LCD-skärm.

Konstruktion

ATMega 32

Processorn som konsolen använder är av typen ATMega 32. De data som har varit av intresse för projektet är att den har 2kB minne och 40 pinnar fördelade på fyra portar, A -D.

Port A används för att skicka data från processorn till skärmen. Datan består antingen av x- och y-adresser för att ange koordinater, eller information om vilka pixlar på skärmen som ska vara tända eller släckta.

Port B används för att skicka instruktioner från processorn till skärmen. Instruktionerna anger hur skärmen ska hantera den data som kommer från port A. Exempel på sådana instruktionsdata är vilken skärmhalva datan från port A ska skickas till, om det är en adress som ska sparas eller om det är data som ska skrivas ut på skärmen.

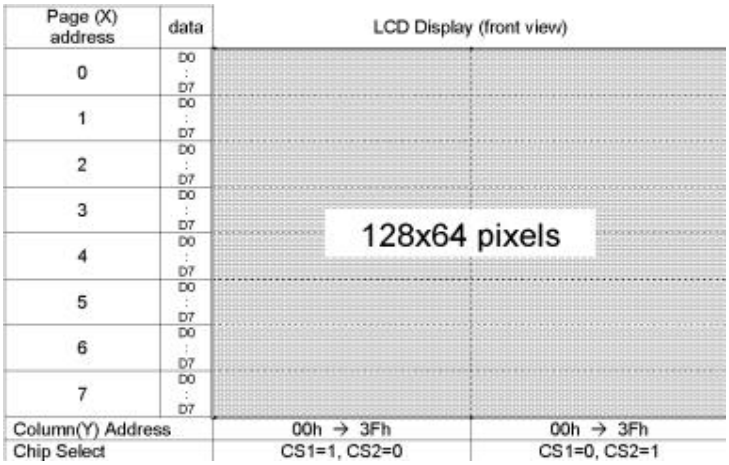
Port C reserverades åt JTAG.

Port D användes för att ta emot information från knapparna.

LCD-skärm

En 128x64 pixel monokrom LCD-skärm användes för att presentera spelet. Skärmen är uppdelad i vänster och höger skärmhalva, båda av storlek 64x64 pixlar. Dessa hanteras separat. I varje skärmhalva finns 64 stycken y-kolonner, dvs en för varje pixel. I x-led är skärmhalvan uppdelad i åtta

stycken register, och där varje register består av åtta pixlar. Således kan pixlarna i x-led inte hanteras individuellt utan alltid i register om åtta.



Knappar

Knapparna är av enkel modell. När de trycks ned släpper de igenom en spänning till processorn, då detta sker blir pinnen som är kopplade till knappen hög. Då knappen inte är nedtryckt är pinnen låg. Signalen skickas dels direkt in till processorns port D och dels skickas den till OR-grinden(se nedan).

JTAG

JTAG-en används för att koppla ihop processorn med programmet AVR Studio. I AVR Studio skrevs och testades programkoden.

74HC32

För att kunna läsa av om någon knapparna är nedtryckt används en OR-grind. Samtliga knappar är kopplade till grinden och om någon av dem trycks ned skickas en signal till processorn som då registrerar detta och sedan läser av vilken av knapparna som är nedtryckt.

Resistorer

Samtliga knappar är kopplade till jord, via en resistor. Resistorernas uppgift är att se till spänningen när knapparna ej är nedtryckta blir 0V, dvs insignalen blir låg.

Genomförande

Uppstart

Efter den första veckans förberedande föreläsningar och laborationer påbörjades projektet i vecka två. Gruppen valde efter utförliga och djuplodande diskussioner att utveckla en spelkonsol.

Det första steget var att upprätta en kravspecifikation samt att identifiera vilka komponenter som skulle ingå i spelkonsolen. Därefter togs ett kopplingsdiagram fram, vilket i grova drag visade spelkonsolens uppbyggnad.

Konstruktion av spelkonsolen

När väl kopplingsdiagrammet fanns på plats rekvirerades nödvändiga komponenter och den utrustning som krävdes för konstruktion av spelkonsolen. Därefter påbörjades en period av intensivt lödande och virande. Processorn kopplades samman med knapparna och skärmen, den logiska grinden byggdes upp och slutligen så kopplades spänningen på.

Då den fysiska spelkonsolen fanns på plats vidtog arbetet med att få kontroll över de olika komponenterna och få dem att interagera, samt att utveckla själva spelet. Vi insåg snabbt att denna del av projektet skulle bli en verklig utmaning.

Inledningsvis låg problemet i att kunna ta hand om de signaler som skickades från knapparna till processorn, osäkerheten låg i hur man översätter ettor och nollor till C-kod. När vi väl behärskade detta påbörjades arbetet med att skicka signaler från processorn till skärmen. Att kunna ta hand om dessa signaler samt att bemästra skärmen och dess funktioner skulle visa sig särskilt tidskrävande. Det stora problemet låg i just kommunikationen mellan processorn och skärmen. För att kunna tända och släcka pixlar på skärmen måste dels information skickas angående vilken skärmhalva som operationen ska utföras i, dels måste en y-kolonn och ett x-register specificeras. Därefter måste data skickas som anger vilka av de åtta pixlarna i det angivna x-registret som skall vara tända respektive släckta. Att få detta att fungera tog mycket tid i anspråk, inte minst då vi efter en tids arbete insåg att kopplingen mellan processorn och skärmen var felaktig och inte överrensstämde med ritningarna.

Snake

När vi väl hade kontroll på komponenterna påbörjades arbetet med själva spelet. Skärmen delades upp i två delar där vänster skärmhalva fick innehålla spelplanen medan höger skärmhalva fick innehålla en poängräknare.

I koden valde vi att representera spelplanen som en 8x64 matris, dvs en kopia av skärmens uppbyggnad. Ormen valde vi att representera med en 2x120 matris, där kolonnerna utgör koordinater för de pixlar på spelplanen där ormen finns. I första raden finns x-värden och i andra raden finns de tillhörande y-värdena. De 120 kolonnerna motsvarar den maximala längden som ormen kan få under spelets gång. När spelet körs så uppdateras koordinaterna i denna matris och ritas sedan ut på skärmen. Detta motsvarar ormens förflyttning på skärmen. Även äpplet representeras av en matris, i detta fall en 2x1-matris där första raden innehåller x-värdet och andra raden innehåller y-värdet. Äpplen placeras ut, med hjälp av en egendesignad slumpgenerator, då spelet påbörjas, samt då ormen äter ett äpple.

Då spelet körs känner programmet av om någon av de fem knapparna är nedtryckta och beroende på vilken som trycks ned körs till den knappen hörande metoder. Det som händer då ormen förflyttar sig ett steg är att matrisen innehållandes ormens koordinater uppdateras. Den nya koordinaten som läggs till, dvs ”nästa steg” jämförs dels med koordinaterna för väggarna och ormens övriga koordinater för att se om spelet ska avbrytas, dels jämförs den med koordinaten för äpplet för att se om en poäng ska adderas till resultatet och ett nytt äpple placeras ut.

Utvecklingen av spelet har gått relativt smidigt. Problemen vi haft har huvudsakligen varit kopplade till vår ovana att skriva C-kod och då framförallt det faktum att C inte är objektsorienterad, något som vi är vana vid från tidigare JAVA-kurser. Ytterligare en faktor som vållade en del problem var processorns begränsade prestanda. Den första versionen av spelet blev alltför stor och översteg

kraftigt kapaciteten hos processorn. Detta var en begränsning vi inte tidigare reflekterat över men som nu fick oss att förstå vikten av resurseffektiv kodning.

Resultat

Vår färdiga prototyp överensstämmer väl med det vi hade i åtanke vid projektets början. Den färdiga prototypen fungerar bra och uppfyller alla de krav som finns listade i kravspecifikationen.

Diskussion

Efter flera veckors intensivt arbete så har vi lyckats konstruera en konsol och ett spel som vi är mycket nöjda med. Inledningsvis fanns en känsla av att våra kunskaper rörande allt från digitalteknik till C-programmering var alltför bristfälliga för att vi skulle kunna konstruera en fungerande spelkonsol. Det har dock visat sig att just denna okunskap har varit en av de faktorer som gjort kursen så pass intressant och lärorik som den faktiskt varit. Problemen som har dykt upp under arbetes gång har löst ett efter ett och successivt har vi lärt oss att bemästra både hård- och mjukvara.

Vi uppskattade verkligen att få bygga någonting från grunden. Det är inte ofta man får använda händerna på det sättet i vår utbildning. Att ha ett stort projekt över lång tid där man själv får disponera tiden uppskattade vi mycket.

Referenser

1. Datablad för processorn
   http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf
   (Notera att konsolen har en processor av typ ATmeg32 och databladet är för en processor av typ ATMega16, de två processorerna är dock identiska frånsett att ATmega32 har 32kb minne medan ATMega16 endast har 16kb minne.)

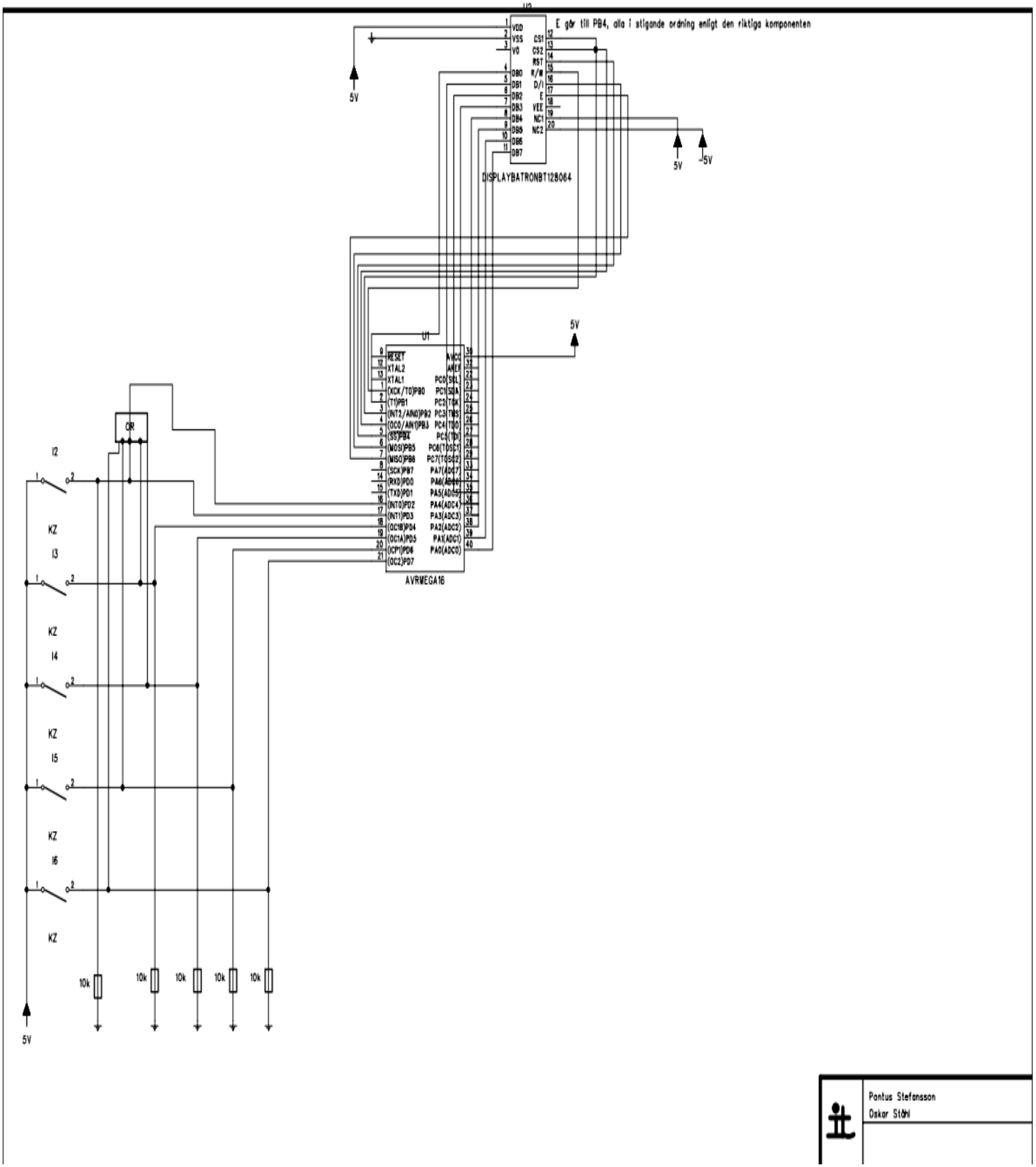
2. Datablad för LCD-skärmen
   http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/GDM12864C.pdf

3. Datablad för OR-grinden
   http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Logik/74HC/74HC32.pdf

4. Brian W. Kernigan, Dennis M.Ritchie, The C programming language, second edition, Prentice Hall Software Series.

Bilagor

Kopplingsschema



Pontus Stefansson
Oskar Ståhl

Källkod

```
#include <avr/io.h>

unsigned short int snake[2][120];
unsigned short int board[8][64];
unsigned short int apple[2][1];
unsigned short int dir;
unsigned short int knappar;
unsigned short int length;
unsigned short int count;
unsigned long int next;
unsigned long int score;


void main()
{
		startUp();
		erase();
		startGame();
}


/*Sätter A- och B-portarna till output och D-porten till input*/
void startUp()
{
		DDRD = 0x00;
		DDRA = 0xFF;
		DDRB = 0xFF;
		startLCD();
}


void startLCD()
{
		PORTA = 0x3F;
		PORTB = 0x29;  //startar vänstra skärmen
		toggleE();

		PORTB = 0x19; //startar högra skärmen
		toggleE();
}


/*Startar upp spelet*/
void startGame()
{
		printSnake();
		while(1)
		{
				knappar=PIND & 0xF8;
				if(knappar == 0x80)
```

```
                        {
                                next = 1;
                                score = 0;
                                printBorder();
                                printZero(15);
                                printZero(35);
                                makeSnake();
                                mainLoop();
                                frozenScreen();
                        }
                }
}


/*Fryser skärmen när spelet är slut*/
void frozenScreen()
{
                while(1)
                        {
                        knappar=PIND & 0xF8;
                        if(knappar == 0x80)
                        {
                                erase();
                                printSnake();
                                delnms(10);
                                break;
                        }
                }
}


/*Växlar E-signalen från hög till låg till hög*/
void toggleE()
{
                PORTB = PORTB | 0x08;
                PORTB = PORTB & 0xF7;
                PORTB = PORTB | 0x08;
}


/*Fördröjer spelet med n millisekunder*/
void delnms(unsigned short int n){
unsigned int z;

  while(n--){
    z=260;
    while(z--);
  }
}


/*Genererar ett slumptal mellan 0 och 63*/
```

```
int rand()
{
		next = next * 1103515245 + 12345;
		return (unsigned int)(next/65536) % 63;
}


/*Skapar ett nytt äpple*/
void makeApple()
{
		apple[0][0] = rand();
		apple[1][0] = rand();

		for(unsigned short int i = 0; i < length; i++)
		{
			if(apple[0][0] == snake[0][i] && apple[1][0] == snake[1][i])
			{
				makeApple();
				return;
			}
		}
		calcCoord(apple[0][0], apple[1][0], 100, 100);
}


/*Ritar ut ormen vid spelets början*/
void makeSnake()
{
		length = 4;
		snake[0][0] = 32;
		snake[1][0] = 20;

		board[4][20] = 0;
		snake[0][1] = 32;
		snake[1][1] = 21;

		board[4][21] = 1;
		snake[0][2] = 32;
		snake[1][2] = 22;
		board[4][22] = 1;

		snake[0][3] = 32;
		snake[1][3] = 23;

		board[4][23] = 1;
		snake[0][4] = 32;
		snake[1][4] = 24;

		board[4][24] = 1;
}
```

```
/*Gör om en koordinat från 64x64 till korrekt x- och y-register*/
void calcCoord(unsigned short int x, unsigned short int y,unsigned short int x2,unsigned short int y2)
{
		unsigned short int data;
		unsigned short int data2;
		unsigned short int xReg = x/8;
		unsigned short int xPixel = x%8;
		unsigned short int x2Reg = x2/8;
		unsigned short int x2Pixel = x2%8;


		if(xPixel == 0) {
				data = 0x01;
		}else if (xPixel == 1) {
				data = 0x02;
		}else if (xPixel == 2) {
				data = 0x04;
		}else if (xPixel == 3) {
				data = 0x08;
		}else if (xPixel == 4) {
				data = 0x10;
		}else if (xPixel == 5) {
				data = 0x20;
		}else if (xPixel == 6) {
				data = 0x40;
		}else if (xPixel == 7) {
				data = 0x80;
		}
		if(x2 != 100 && y2 != 100){
				if(x2Pixel == 0) {
						data2 = 0xFE;
				}else if (x2Pixel == 1) {
						data2 = 0xFD;
				}else if (x2Pixel == 2) {
						data2 = 0xFB;
				}else if (x2Pixel == 3) {
						data2 = 0xF7;
				}else if (x2Pixel == 4) {
						data2 = 0xEF;
				}else if (x2Pixel == 5) {
						data2 = 0xDF;
				}else if (x2Pixel == 6) {
						data2 = 0xBF;
				}else if (x2Pixel == 7) {
						data2 = 0x7F;
				}
				board[x2Reg][y2] = board[x2Reg][y2] & data2;
		}
		board[xReg][y] = board[xReg][y] | data;

}
```

```
/*Kollar om ormen krockat med en vägg eller sig själv*/
int gameOver()
{
		unsigned short int headXCoord = snake[0][length];
		unsigned short int headYCoord = snake[1][length];

		if(headXCoord < 0|| headYCoord < 0 || headXCoord > 63|| headYCoord > 63)
		{
			return 0;
		}

		for(unsigned short int i = 0; i < length; i++)
		{
			if(headXCoord == snake[0][i] && headYCoord == snake[1][i])
			{
				return 0;
			}
		}
		return 1;
}


/*Kollar om ormen tagit ett äpple*/
int gotApple()
{
		if(snake[0][length] == apple[0][0] && snake[1][length] == apple[1][0])
		{
			return 0;
		}
		return 1;
}


/*Förlänger ormen med en pixel*/
void growSnake()
{
		if(length < 118)
		{
		moveWhenGrow();
		}
}


/*Förflyttningsfunktion samma runda som ormen växer*/
void moveWhenGrow()
{
		switch(dir)
		{
			case 0:
			break;
```

```
                        case 1: //upp
                                length++;
                                snake[0][length] = snake[0][length - 1] - 1;
                                snake[1][length] = snake[1][length - 1];
                                calcCoord(snake[0][length],snake[1][length], snake[0][0],
snake[1][0]);
                        break;


                        case 2: //höger
                                length++;
                                snake[0][length] = snake[0][length - 1];
                                snake[1][length] = snake[1][length - 1] + 1;
                                calcCoord(snake[0][length],snake[1][length], snake[0][0],
snake[1][0]);
                        break;


                        case 3: //ned
                                length++;
                                snake[0][length] = snake[0][length - 1] + 1;
                                snake[1][length] = snake[1][length - 1];
                                calcCoord(snake[0][length],snake[1][length], snake[0][0],
snake[1][0]);
                        break;


                        case 4:         //vänster
                                length++;
                                snake[0][length] = snake[0][length - 1];
                                snake[1][length] = snake[1][length - 1] - 1;
                                calcCoord(snake[0][length],snake[1][length], snake[0][0],
snake[1][0]);
                        break;
                }
}


/*Förflyttningsfunktion en vanlig runda*/
void move()
{

                signed short int temp;
                switch(dir)
                {
                        case 0: //paus
                        break;


                        case 1: //upp
                                temp = snake[0][length] - 1;
```

```
                                        for(unsigned short int i = 1; i <= length; i++)
                                        {
                                                snake[0][i-1] = snake[0][i];
                                                snake[1][i-1] = snake[1][i];
                                        }
                                        snake[0][length] = temp;
                                        calcCoord(snake[0][length],snake[1][length], snake[0][0],
snake[1][0]);
                                break;


                                case 2: //höger


                                        temp = snake[1][length] + 1;
                                        for(unsigned short int m= 1; m <= length; m++)
                                        {
                                                snake[1][m-1] = snake[1][m];
                                                snake[0][m-1] = snake[0][m];
                                        }
                                        snake[1][length] = temp;
                                        calcCoord(snake[0][length],snake[1][length], snake[0][0],
snake[1][0]);
                                break;


                                case 3: //ned
                                        temp = snake[0][length] + 1;
                                        for(unsigned short int i = 1; i <= length; i++)
                                        {
                                                snake[0][i-1] = snake[0][i];
                                                snake[1][i-1] = snake[1][i];

                                        }
                                        snake[0][length] = temp;
                                        calcCoord(snake[0][length],snake[1][length], snake[0][0],
snake[1][0]);
                                break;


                                case 4:         //vänster
                                temp = snake[1][length] - 1;
                                for(unsigned short int n = 1; n <= length; n++)
                                {
                                        snake[1][n-1] = snake[1][n];
                                        snake[0][n-1] = snake[0][n];
                                }
                                snake[1][length] = temp;
                                calcCoord(snake[0][length],snake[1][length], snake[0][0], snake[1][0]);
                                break;
                        }
}
```

```
/*Skriver något till vänstra skärmhalvan*/
void writeLeft(unsigned short int x, unsigned short int y, unsigned short int data)
{
		PORTB = 0x29;

		PORTA = 0x40 + y; //y
		toggleE();

		PORTA = 0xB8 + x; //x
		toggleE();

		PORTB = 0x2B; //writemode
		PORTA = data;
		toggleE();

		PORTB = 0x29;
		PORTA = 0x00;
}


/*Skriver något till högra skärmhalvan*/
void writeRight(unsigned short int x, unsigned short int y, unsigned short int data)
{
		PORTB = 0x19;

		PORTA = 0x40 + y; //y
		toggleE();

		PORTA = 0xB8 + x; //x
		toggleE();

		PORTB = 0x1B; //writemode
		PORTA = data;
		toggleE();

		PORTB = 0x19;
		PORTA = 0x00;
}


/*Skriver ut "Score" på vänstra skärmhalvan*/
void printBorder()
{
		erase();
		for(unsigned short int x = 0; x <= 7; x++)
		{
			for(unsigned short int y = 0; y <= 63; y++)
			{
				board[x][y] =0x00;
			}
```

```
}
for(unsigned short int x = 0; x <= 7; x++)
{
		writeRight(x, 0, 0xFF);
}
for(short int y = 15; y <=20; y++) /*SkrivS	*/
{
		writeRight(1, y, 0x41);
		writeRight(2, y, 0x08);
}
writeRight(1, 15, 0x7F);
writeRight(1, 20, 0xC0);
writeRight(2, 20, 0x0F);                                        /*SkrivS*/

for(short int y = 23; y <= 28; y++)             /*SkrivC*/
{
		writeRight(1, y, 0x01);
		writeRight(2, y, 0x08);
}
writeRight(1, 23, 0xFF);
writeRight(2, 23, 0x0F);                                        /*SkrivC*/

for(short int y = 31; y <= 36; y++)             /*SkrivO*/
{
		writeRight(1, y, 0x01);
		writeRight(2, y, 0x08);
}
writeRight(1, 31, 0xFF);
writeRight(1, 36, 0xFF);
writeRight(2, 31, 0x0F);
writeRight(2, 36, 0x0F);

writeRight(1,39,0xFF); /*SkrivR*/
writeRight(2,39,0x0F);
writeRight(1,45,0x7F);

for(short int y = 41; y <=44; y++)
{
		writeRight(1, y, 0x41);
}

writeRight(1,40,0xC1);
writeRight(1,41,0xC1);
writeRight(2,42,0x01);
writeRight(2,43,0x02);
writeRight(2,44,0x04);
writeRight(2,45,0x08);

for(short int y = 48; y <=53; y++) /*SkrivE	*/
{
		writeRight(1, y, 0x41);
		writeRight(2, y, 0x08);
```

```
		}
		writeRight(1, 48, 0xFF);

		writeRight(2, 48, 0x0F);



}


/*Suddar allt på skärmen*/
void erase()
{
		for(unsigned short int x = 0; x <= 7; x++)
		{
			for(unsigned short int y = 0; y <= 63; y++)
			{
				writeLeft(x,y,0x00);
				writeRight(x,y,0x00);
			}
		}
}


/*Uppdaterar alla förändringar på spelplanen*/
void writeBoard()
{
		for(short int x = 0; x < 8; x++){
			for(short int y = 0; y < 64; y++){
				writeLeft(x,y,board[x][y]);
			}
		}
}

/*Funktionerna nedan skriver ut siffrorna 0-9*/
void printZero(unsigned short int coord)
{

		for(unsigned short int x = 4; x <= 7; x++)
		{
			for(unsigned short int y = coord; y <= coord + 16; y++)
			{
				writeRight(x,y,0x00);
			}
		}

		for(short int y=coord; y<=coord+15; y++)	//horisontellt
		{
			writeRight(4,y,0x03);
			writeRight(6,y,0xC0);
		}
```

```
		for(short int x=4; x<7; x++)                                    //vertikalt
		{
			writeRight(x,coord,0xFF);
			writeRight(x,coord+1,0xFF);
			writeRight(x,coord+15,0xFF);
			writeRight(x,coord+16,0xFF);
		}
}


void printOne(unsigned short int coord)
{

		for(unsigned short int x = 4; x <= 7; x++)
		{
			for(unsigned short int y = coord; y <= coord + 16; y++)
			{
				writeRight(x,y,0x00);
			}
		}
		for(short int x = 4; x < 7; x++)
		{
			writeRight(x, coord + 15, 0xFF);
			writeRight(x, coord + 16, 0xFF);
		}
}


void printTwo(unsigned short int coord)
{

		for(unsigned short int x = 4; x <= 7; x++)
		{
			for(unsigned short int y = coord; y <= coord + 16; y++)
			{
				writeRight(x,y,0x00);
			}
		}
		for(short int i = 0; i <= 15; i++)
		{
			writeRight(4, coord + i, 0x03);
			writeRight(5, coord + i, 0x18);
			writeRight(6, coord + i, 0xC0);
		}
		writeRight(4, coord + 14, 0xFF);
		writeRight(4, coord + 15, 0xFF);
		writeRight(5, coord + 14, 0x1F);
		writeRight(5, coord + 15, 0x1F);
		writeRight(5, coord, 0xF8);
		writeRight(5, coord + 1, 0xF8);
		writeRight(6, coord, 0xFF);
```

```
		writeRight(6, coord + 1, 0xFF);

}


void printThree(unsigned short int coord)
{
		for(unsigned short int x = 4; x <= 7; x++)
		{
			for(unsigned short int y = coord; y <= coord + 16; y++)
			{
				writeRight(x,y,0x00);
			}
		}
		for(short int y = 0; y <= 13; y++)
		{
			writeRight(4, coord + y, 0x03);
			writeRight(5, coord + y, 0x18);
			writeRight(6, coord + y, 0xC0);
		}
		writeRight(4, coord + 14, 0xFF);
		writeRight(4, coord + 15, 0xFF);
		writeRight(5, coord + 14, 0xFF);
		writeRight(5, coord + 15, 0xFF);
		writeRight(6, coord + 14, 0xFF);
		writeRight(6, coord + 15, 0xFF);
}


void printFour(unsigned short int coord)
{
		for(unsigned short int x = 4; x <= 7; x++)
		{
			for(unsigned short int y = coord; y <= coord + 16; y++)
			{
				writeRight(x,y,0x00);
			}
		}
		for(short int y = 2; y <= 13; y++)
		{
			writeRight(5, coord + y, 0x18);
		}
		writeRight(4, coord, 0xFF);
		writeRight(4, coord + 1, 0xFF);
		writeRight(4, coord + 14, 0xFF);
		writeRight(4, coord + 15, 0xFF);
		writeRight(5, coord, 0x1F);
		writeRight(5, coord + 1, 0x1F);
		writeRight(5, coord + 14, 0xFF);
		writeRight(5, coord + 15, 0xFF);
		writeRight(6, coord + 14, 0xFF);
		writeRight(6, coord + 15, 0xFF);
```

```
}


void printFive(unsigned short int coord)
{
		for(unsigned short int x = 4; x <= 7; x++)
		{
			for(unsigned short int y = coord; y <= coord + 16; y++)
			{
				writeRight(x,y,0x00);
			}
		}

		for(short int i = 0; i <= 15; i++)
		{
			writeRight(4, coord + i, 0x03);
			writeRight(5, coord + i, 0x18);
			writeRight(6, coord + i, 0xC0);
		}
		writeRight(4, coord, 0xFF);
		writeRight(4, coord + 1, 0xFF);
		writeRight(5, coord, 0x1F);
		writeRight(5, coord + 1, 0x1F);
		writeRight(5, coord + 14, 0xF8);
		writeRight(5, coord + 15, 0xF8);
		writeRight(6, coord + 14, 0xFF);
		writeRight(6, coord + 15, 0xFF);
}


void printSix(unsigned short int coord)
{
		for(unsigned short int x = 4; x <= 7; x++)
		{
			for(unsigned short int y = coord; y <= coord + 16; y++)
			{
				writeRight(x,y,0x00);
			}
		}
		for(short int i = 0; i <= 15; i++)
		{
			writeRight(5, coord + i, 0x18);
			writeRight(6, coord + i, 0xC0);
		}
		writeRight(4, coord, 0xFF);
		writeRight(4, coord + 1, 0xFF);
		writeRight(5, coord, 0xFF);
		writeRight(5, coord + 1, 0xFF);
		writeRight(5, coord, 0xFF);
		writeRight(5, coord + 1, 0xFF);
		writeRight(5, coord + 14, 0xF8);
		writeRight(5, coord + 15, 0xF8);
```

```
		writeRight(6, coord, 0xFF);
		writeRight(6, coord + 1, 0xFF);
		writeRight(6, coord + 14, 0xFF);
		writeRight(6, coord + 15, 0xFF);
}


void printSeven(unsigned short int coord)
{
		for(unsigned short int x = 4; x <= 7; x++)
		{
			for(unsigned short int y = coord; y <= coord + 16; y++)
			{
				writeRight(x,y,0x00);
			}
		}
		for(short int i = 0; i <= 15; i++)
		{
			writeRight(4, coord + i, 0x03);
		}
		for(short int x = 4; x < 7; x++)
		{
			writeRight(x, coord + 15, 0xFF);
			writeRight(x, coord + 16, 0xFF);
		}
}


void printEight(unsigned short int coord)
{
		for(unsigned short int x = 4; x <= 7; x++)
		{
			for(unsigned short int y = coord; y <= coord + 16; y++)
			{
				writeRight(x,y,0x00);
			}
		}

		for(short int i = 0; i <= 15; i++)
		{
			writeRight(4, coord + i, 0x03);
			writeRight(5, coord + i, 0x18);
			writeRight(6, coord + i, 0xC0);
		}
		writeRight(4, coord, 0xFF);
		writeRight(4, coord + 1, 0xFF);
		writeRight(4, coord + 14, 0xFF);
		writeRight(4, coord + 15, 0xFF);
		writeRight(5, coord, 0xFF);
		writeRight(5, coord + 1, 0xFF);
		writeRight(5, coord + 14, 0xFF);
		writeRight(5, coord + 15, 0xFF);
```

```
		writeRight(6, coord, 0xFF);
		writeRight(6, coord + 1, 0xFF);
		writeRight(6, coord + 14, 0xFF);
		writeRight(6, coord + 15, 0xFF);
}


void printNine(unsigned short int coord)
{
		for(unsigned short int x = 4; x <= 7; x++)
		{
			for(unsigned short int y = coord; y <= coord + 16; y++)
			{
				writeRight(x,y,0x00);
			}
		}

		for(short int i = 0; i <= 15; i++)
		{
			writeRight(4, coord + i, 0x03);
			writeRight(5, coord + i, 0x18);
			writeRight(6, coord + i, 0xC0);
		}
		writeRight(4, coord, 0xFF);
		writeRight(4, coord + 1, 0xFF);
		writeRight(4, coord + 14, 0xFF);
		writeRight(4, coord + 15, 0xFF);
		writeRight(5, coord, 0x1F);
		writeRight(5, coord + 1, 0x1F);
		writeRight(5, coord + 14, 0xFF);
		writeRight(5, coord + 15, 0xFF);
		writeRight(6, coord + 14, 0xFF);
		writeRight(6, coord + 15, 0xFF);
}


/*Huvudloopen när spelet är igång*/
void mainLoop()
{
		dir = 2;
		count = 0;
		makeApple();
		while(1)
		{

			if(count == 10 && length < 118 && dir!=0) {
				growSnake();
				count = 0;
			} else {
				move();

			}
```

```
knappar=PIND & 0xF8;
if(gameOver() == 0)
{
                break;
}
if(gotApple() == 0)                                    //Fixa innan kör
{
                score++;
                printScore();
                if(length < 118)
                {
                growSnake();
                }
                makeApple();
}
delnms(5);

switch(knappar)
{

case 0x08:  //Upp-knapp
if(dir!=3){
dir = 1;

}
break;

case 0x10:      //Höger-knapp
if(dir!=4){
dir = 2;

}
break;

case 0x20:  //Ner-knapp
if(dir!=1){
dir = 3;

}
break;

case 0x40:      //Vänster-knapp
if(dir!=2){
dir = 4;

}
break;

case 0x80:      //Svarta knappen
dir = 0;
break;
```

```
                        }
                        writeBoard();
                        count++;
                }
}


/*Skriver ut resultatet*/
void printScore()
{
                unsigned short int firstNbr = score/10;
                unsigned short int secondNbr = score % 10;

                if(firstNbr == 0)                                       //Första siffran
                {
                        printZero(15);
                }
                if(firstNbr == 1)
                {
                        printOne(15);
                }
                if(firstNbr == 2)
                {
                        printTwo(15);
                }
                if(firstNbr == 3)
                {
                        printThree(15);
                }
                if(firstNbr == 4)
                {
                        printFour(15);
                }
                if(firstNbr == 5)
                {
                        printFive(15);
                }
                if(firstNbr == 6)
                {
                        printSix(15);
                }
                if(firstNbr == 7)
                {
                        printSeven(15);
                }
                if(firstNbr == 8)
                {
                        printEight(15);
                }
                if(firstNbr == 9)
                {
                        printNine(15);
```

```
		}


		if(secondNbr == 0)					//Andra siffran
		{
			printZero(35);
		}
		if(secondNbr == 1)
		{
			printOne(35);
		}
		if(secondNbr == 2)
		{
			printTwo(35);
		}
		if(secondNbr == 3)
		{
			printThree(35);
		}
		if(secondNbr == 4)
		{
			printFour(35);
		}
		if(secondNbr == 5)
		{
			printFive(35);
		}
		if(secondNbr == 6)
		{
			printSix(35);
		}
		if(secondNbr == 7)
		{
			printSeven(35);
		}
		if(secondNbr == 8)
		{
			printEight(35);
		}
		if(secondNbr == 9)
		{
			printNine(35);
		}

}

/*Funktionerna nedan skriver ut bokstäverna S N A K E*/
void printS()
{
		for(unsigned int y = 5; y <= 25; y++) {
					writeLeft(1,y,0x30);
					writeLeft(2,y,0xC0);
```

```
                        }
                        for(unsigned int y = 5; y <= 24; y++) {
                                writeLeft(4,y,0x03);
                        }
                        writeLeft(1, 5,0xF0);
                        writeLeft(1, 6,0xF0);
                        writeLeft(2, 5,0xFF);
                        writeLeft(2, 6,0xFF);
                        writeLeft(3, 24,0xFF);
                        writeLeft(3, 25,0xFF);
                        writeLeft(4, 24,0x03);
                        writeLeft(4, 25,0x03);

}

void printN() {

                writeLeft(1, 29,0xF0);
                writeLeft(1, 30,0xF0);
                writeLeft(2, 29,0xFF);
                writeLeft(2, 30,0xFF);
                writeLeft(3, 29,0xFF);
                writeLeft(3, 30,0xFF);
                writeLeft(4, 29,0x03);
                writeLeft(4, 30,0x03);

                writeLeft(1 ,48,0xF0);
                writeLeft(1 ,49,0xF0);
                writeLeft(2, 48,0xFF);
                writeLeft(2, 49,0xFF);
                writeLeft(3, 48,0xFF);
                writeLeft(3, 49,0xFF);
                writeLeft(4, 48,0x03);
                writeLeft(4, 49,0x03);

                for(unsigned short int n = 0; n < 2; n++)
                {
                        writeLeft(2, 31,0x07);
                        writeLeft(2, 32+n,0x1C);
                        writeLeft(2, 34+n,0x70);
                        writeLeft(2, 36+n,0xC0);
                        writeLeft(3,36+n,0x01);
                        writeLeft(3,38+n,0x07);
                        writeLeft(3,40+n,0x1C);
                        writeLeft(3,42+n,0x70);
                        writeLeft(3,44+n,0xC0);
                        writeLeft(4,44+n,0x01);
                        writeLeft(4,46+n,0x03);
                }

}
```

```
void printA() {

		for(unsigned short int n = 0; n < 2; n++) {
		writeLeft(1, 51+n,0xF0);			//ritar första benet
		writeLeft(2,51+n,0xFF);
		writeLeft(3, 51+n,0xFF);
		writeLeft(4, 51+n,0x03);

		writeRight(1, 5+n,0xF0);			//ritar andra benet
		writeRight(2,5+n,0xFF);
		writeRight(3, 5+n,0xFF);
		writeRight(4, 5+n,0x03);

		}
		for(unsigned int n = 0; n <= 10; n++) {  //ritar tak
				writeLeft(1, 53+n, 0x30);
				writeLeft(2, 53+n, 0x30);
		}
		for(unsigned int n = 0; n <= 4; n++) {		//ritar mitten
				writeRight(1, n, 0x30);
				writeRight(2, n, 0x30);
		}
}

void printK() {

		writeRight(1, 9,0xF0);
		writeRight(1, 10,0xF0);

		writeRight(2,9,0xFF);
		writeRight(2,10,0xFF);

		writeRight(3, 9,0xFF);
		writeRight(3, 10,0xFF);

		writeRight(4, 9,0x03);
		writeRight(4, 10,0x03);
		for(unsigned short int n = 0; n < 2; n++)
		{

				writeRight(2,11+n,0xC0);
				writeRight(2,13+n,0xF0);
				writeRight(2,15,0x30);
				writeRight(2,16+n,0x30);
				writeRight(2,17,0x3C);
				writeRight(2,18+n,0x0C);
				writeRight(2,20,0x0C);
				writeRight(2,21,0x0F);
				writeRight(2,22+n,0x03);
				writeRight(2,24, 0x03);
				writeRight(1,24, 0xC0);
				writeRight(1,25,0xC0);
```

```
                writeRight(1,26+n,0xC0);
                writeRight(1,28,0xF0);
                writeRight(1,29+n,0x30);

                writeRight(3, 13+n, 0x03);
                writeRight(3, 15+n, 0x03);
                writeRight(3, 17, 0x0F);
                writeRight(3, 18+n, 0x0C);
                writeRight(3, 20, 0x3C);
                writeRight(3, 21+n, 0x30);
                writeRight(3, 23, 0xF0);
                writeRight(3, 24+n, 0xC0);
                writeRight(3, 26+n, 0xC0);
                writeRight(4, 27, 0x03);
                writeRight(4, 28+n, 0x03);

        }
}

void printE() {

        writeRight(1,33,0xF0);
        writeRight(1,34,0xF0);

        writeRight(2,33,0xFF);
        writeRight(2,34,0xFF);

        writeRight(3,33,0xFF);
        writeRight(3,34,0xFF);

        writeRight(4,33,0x03);
        writeRight(4,34,0x03);


        for(int n = 0; n <= 17; n++) {
                writeRight(1, 35 + n, 0x30);
                writeRight(4, 35 + n, 0x03);
        }

        for(int y = 0; y <= 14; y++) {
                writeRight(2, 35 + y, 0x30);
        }
}


/*Skriver ut ordet SNAKE*/
void printSnake()
{
        printS();
        printN();
        printA();
        printK();
```

```
                printE();
}
```