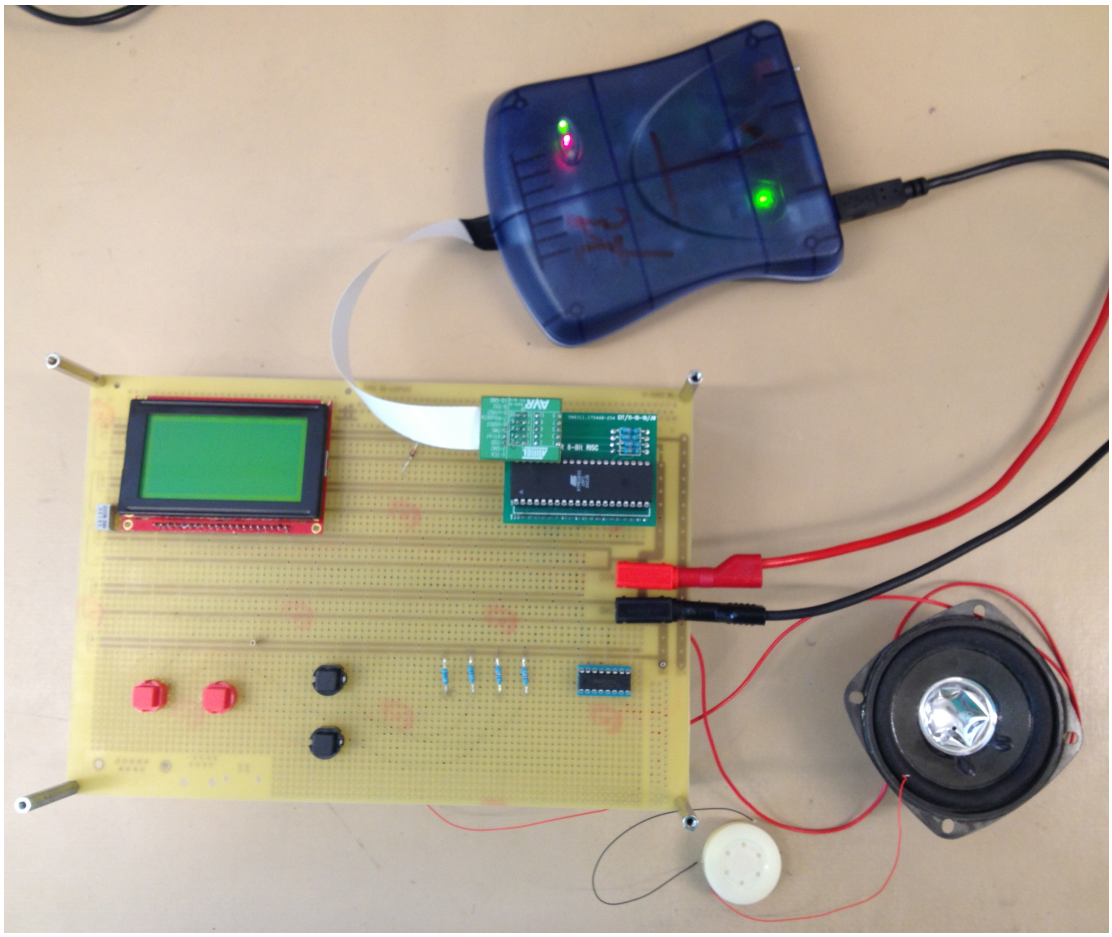
Brick Buster

Henrik Nilsson, Filip Andersson och Martin Ernemar



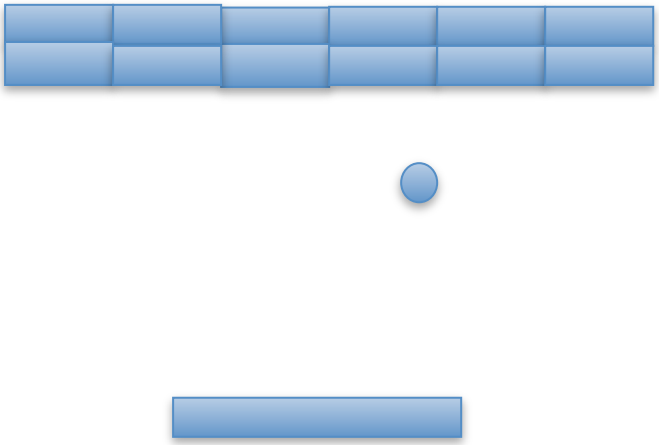
Sammanfattning

Projektet som genomfördes bestod av att konstruera och koppla samman ett antal hårdvarukomponenter och därefter skriva den programkod som krävs till ett enklare dataspel kallat Brick Buster. Spelet går ut på att spelaren ska studsa en boll mellan en plattform och ett antal block för att på så sätt ta sönder blocken och samla poäng. Den mest tidskrävande aktiviteten var att programmera själva spelet och metoderna för att presentera detta på rätt sätt för spelaren. Utfallet av projektet blev ett fullt fungerande spel som visualiseras på en LCD-skärm och exekveras på en processor på 8 MHz och spelaren styr spelet via fyra knappar.

Innehållsförteckning

Inledning

Målet med kursen är att driva och genomföra ett projekt från idé till en färdig prototyp. I vårt fall föll valet på att bygga en enkel spelkonsol och skriva programkoden för att kunna spela ett spel vi valt kalla ”Brick Buster”. Spelidén för ”Brick Buster” bygger på att spelaren kan flytta en plattform, vilken är positionerad längst ner på skärmen, i sidled. Högst upp på skärmen finns ett antal block, som spelaren ska förstöra genom att studsa en boll mellan plattformen och blocken. Se figur 1.



Figur 1 - Illustration av Brick Buster

Kravspecifikation

Följande krav kommer att ställas på design och programvara för spelet:

- Spelet ska visualiseras på en LCD-skärm.
- LCD-skärmen är uppdelad i två halvor, där ena halvan visar information som aktuell poäng och ”highscore” och andra halvan utgör spelskärmen.
- Spelaren ska styra plattformen med två knappar (höger & vänster).
- När ett nytt spel skapats startar spelet när spelaren trycker på en av styrknapparna.
- Spelaren ska kunna pausa och återuppta spelet under spelets gång.
- När spelet är slut ska spelaren kunna välja att starta ett nytt spel.

Eventuella krav (i mån av tid):

- Poängställning ska visas i realtid under spelets gång (1 p/block).
- Spelaren ska kunna välja mellan olika banor.
- Spelaren ska ha tre liv per spelomgång.
- En högtalare spelar upp någon form av ljud under spelets gång.

Komponenter

Detta avsnitt innehåller lite kortare teori om de hårdvarukomponenter som vi använt till vår spelkonsol.

ATMega32

Till vår konsol har vi valt att använda en processor av typen ATMega32 vilken vi körde på frekvensen 8 MHz. Denna processor har 32 kB programminne, 2 kB dataminne och 40 I/O-pinnar uppdelade på fyra olika portar (A-D). Från början använde vi en ATMega16 vilken endast har 16 kB programminne och 1 kB dataminne men eftersom programkoden för spelet i slutändan blev drygt 20 kB behövde vi byta till en ATMega32 istället.

Till vår konsol har vi använt portarna enligt följande:

- Port B är kopplad till LCD-skärmens databuss.
- Port C används dels för kommunikation med J-Tag samt instruktionspinnar på LCD-skärmen.
- Port D är kopplad till fyra knappar samt instruktionspinnar på LCD-skärmen.

LCD-skärm (GDM12864C (Red))

Skärmen har en upplösning på 128x64 pixlar och är uppdelad i två stycken separata skärmhalvor som båda har en upplösning på 64x64. Skärmen har ett inbyggt minne på 16kB och 20 stycken pinnar, av dessa 20 pinnar används åtta stycken för att skicka och ta emot data från processorn. Övriga pinnar används till att specificera hur skärmen ska tolka den data som läggs på databussen, t.ex. sätta på/stänga av skärmen, bestämma vilken skärmhalva som adresseras, ändra kontrasten och om skärmen ska läsa eller skriva information.

Varje skärmhalva hanterar pixlarna i x-led i register om 8 pixlar medan pixlarna i y-led hanteras separat. Detta för att varje koordinatpar ska motsvara åtta bitar (en byte).

Knappar

Knapparna till spelkonsolen utgörs av fyra stycken strömbrytare som är uppdelade i två par. Ett par röda knappar som används för att styra bollplattan och ett par svarta knappar som hanterar menyval.

Resistorer

För att spänningen verkligen ska vara 0V då knapparna inte är nedtryckta så kopplas en resistor på 10k Ohm till varje knapp som sedan kopplas till jord.

En variabel resistor används också för att kunna variera skärmens kontrast. På denna resistor går det att variera motståndet mellan 0 och 10k Ohm för att på så vis ge skärmen en lagom kontrast för den betraktelsevinkel man har.

J-Tag

J-Tag används för att föra över programkoden från en dator till processorn. AVR Studio 4.0 används för att både skriva koden och sedan överföra den med hjälp av J-Tag.

Genomförande

Projektet inleddes med att bestämma vad som skulle byggas och vilka krav vi skulle komma att ställa på vår konstruktion. Dessa krav sammanställdes sedan i en kravspecifikation som återfinns i denna rapport. Nästa steg var att i programmet PowerLogic rita upp ett kopplingsschema, se bilaga 1, för den spelkonsol vi valt att konstruera. Efter att ha fått kopplingsschemat godkänt av vår handledare tilldelades vi en verktygslåda samt de komponenter vi behövde för att kunna börja bygga ihop konsolen enligt kopplingsschemat. Efter en hel del lödande och virande av trådar så var vår konstruktion färdig att testas så att alla kopplingar fungerade som de skulle. För att testa detta användes programmet AVR Studio 4.0 som via J-Tag tog kontroll över konsolen. Genom att direkt klicka på processorns portar i AVR Studio kunde värdet på dessa varieras, helt utan att behöva skriva någon programkod, och på så vis kunde de enskilda komponenternas funktion verifieras. När det var säkerställt att hårdvaran fungerade som planerat började projektets största fas; att skriva programkoden för vår spelkonsol. Med jämna mellanrum skrevs koden över på processorn via J-tagen och provkördes i AVR Studios debug-läge. Till sist stod konsolen färdig med programkoden för det fungerande spelet Brick Buster lagrat på processorn redo att exekveras.

Under projektets gång har allting långt ifrån flutit på smärtfritt. Vi har brottats med en del problem kring olika lösningar i programkoden men också saker som att de hårdvarukomponenter vi blivit tilldelade inte har fungerat som de ska. Till att börja med upptäckte vi när vi höll på att testas hårdvarans funktion att ett ben på den bricka där processorn satt fast inte gav någon signal alls. Felet visade sig vara att det helt enkelt inte fanns någon lödning kring detta ben och därmed ingen koppling till benet på själva processorn. Under testen visade det sig också att LCD-skärmens faktiska egenskaper inte helt stämde överens med de som stod i dess datablad. Efter åtskilliga timmar av felsökande kring varför skärmen inte sattes igång lyckades vi av en slump upptäcka att om man skickade värdet 1 till skärmens reset-pinne så gick skärmen igång, trots att det i databladet stod att den vid normal drift skulle ha värdet 0.

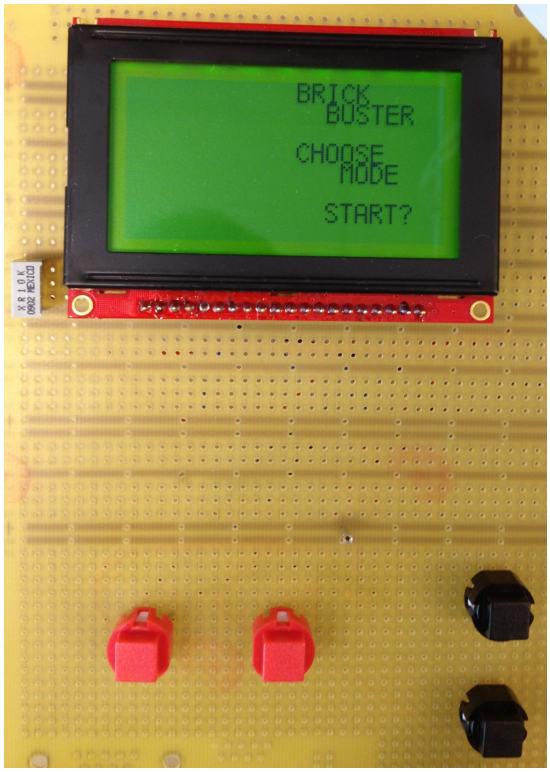
Ytterligare ett problem som fanns med oss under större delen av programmeringsarbetet var att vi inte alltid fick samma resultat när samma kod exekverades flera gånger vilket var både underligt och frustrerande. Detta kom vi dock tillrätta med då vi kom underfund med att vi hela tiden skickade data till de pinnar på processorn som var reserverade för J-Tag vilket kan leda till konstigheter då och då. När detta var åtgärdat blev exekveringen betydligt mer konsekvent dock uppstod nu ett annat problem vilket var att det inte längre gick att läsa av skärmens status. Detta var något vi aldrig kom till rätta med och inte heller vår handledare kunde komma på var felet låg så lösningen blev att vi helt enkelt skrev in en liten tidsfördröjning mellan varje instruktion till skärmen för att försäkra oss om att processorn inte skickade informationen snabbare än vad skärmen kunde ta emot den.

Det var från början även tänkt att vi skulle ha en högtalare som genererade lite enklare ljud under spelets gång. Detta var dock något som aldrig blev verklighet då vi varken själva eller med handledarens hjälp lyckades få någon spänning att ligga kvar över högtalaren och kunde därmed inte generera något ljud. Vi testade

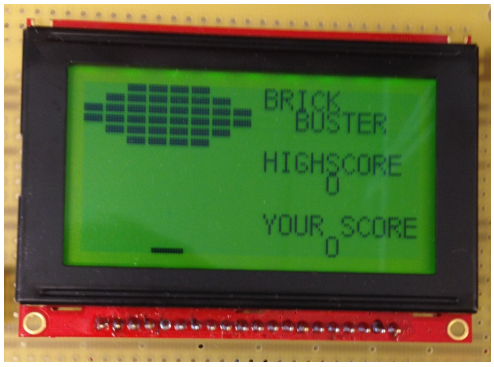
även med en större högtalare med större motstånd men där visade det sig att vi även skulle komma att behöva bygga en förstärkare vilket hade blivit lite väl jobbigt att ta tag i på sluttampen.

Resultat

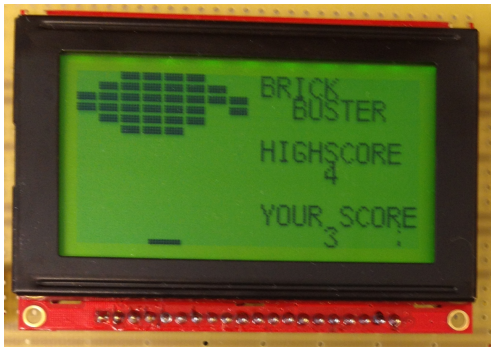
Efter sju veckors arbete blev resultatet en spelkonsol innehållande spelet Brick Buster. När konsolen startas visas spelets meny på skärmen enligt figur 2 där spelaren har möjlighet att välja mellan två olika banor med hjälp av de två svarta knapparna. Spelet startar sedan då spelaren trycker på någon av de röda knapparna vilka flyttar plattformen på bottnen av skärmen till vänster respektive höger. Under spelets gång visas namnet Brick Buster, highscore samt din aktuella score i realtid enligt figur 3 och 4. När spelet är igång har spelaren möjlighet att pausa spelet genom att trycka på den övre svarta knappen och spelet återupptas sedan när spelaren tycker på den undre svarta knappen. Spelet tar slut då spelaren missar att studsa bollen mot plattformen och vänster skärmhalva fryses och menyn skriv åter ut på höger halva, enligt figur 5.



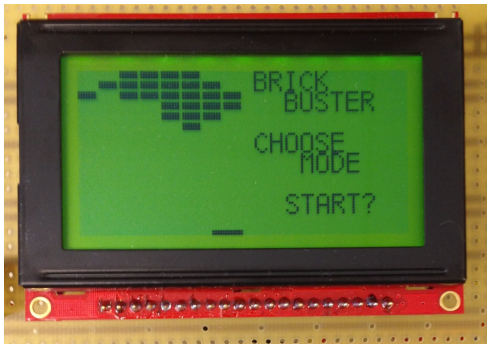
Figur 2 – Bild över skärmen (direkt efter uppstart) samt de fyra knapparna



Figur 3 – Bild över skärmen (precis vid spelstart)



Figur 4 – Bild över skärmen (under spelomgång)



Figur 5 – Bild över skärmen (då spelomgång avslutats)

Diskussion

Detta projekt har fått oss att inse hur mycket tid som verkligen måste läggas på ett projekt från det att man får en idé till man kan presentera en färdig prototyp. En annan mycket viktig insikt som kommit av de åtskilliga problem som vi stött på under projektets gång är att det är väldigt mycket som kan gå fel under ett utvecklingsprojekt. Vidare upplevde vi att en väldigt betydande del av tidsåtgången används för felsökning och ofta av saker som man själv inte helt och hållet kan styra över så som att hårdvarukomponenter inte fungerar som specificerat.

Projektet började väldigt bra för oss inledningsvis, hela gruppen var snabbt överens om vad för funktioner vi ville få med i vårt spel. Ritandet av kopplingsschemat gick väldigt smidigt trots att vi tidigare aldrig använt programmet PowerLogic. Då vår konstruktion inte bestod av så många komponenter var själva byggandet inte allt för svårt eller tidskrävande. Dock innebar det faktum att ingen i gruppen hade gjort något liknande tidigare att en del lödningar och andra kopplingar fick göras om i ett senare skede när det visade sig att vi i början av projektet inte hade lyckades få till dessa tillräckligt bra.

Det som varit den mest krävande delen av vårt projekt är helt klart att skriva programkoden för att kommunicera med hårdvaran och beskriva spelet Brick Buster. Då vi tidigare inte skrivit varken någon kod i C eller överhuvudtaget skrivit programkod för att få hårdvarukomponenterna att kommunicera med varandra och resten av koden blev detta det tyngsta berget att ta sig över. Att sätta sig in i detta tog lång tid och var helt klart det svåraste i detta projekt. Om man som vi tidigare inte har någon direkt erfarenhet av C-programmering så rekommenderar vi att man bara väljer denna typ av projekt om man verkligen har ett genuint intresse för programmering och en naturlig känsla för hur programspråk är uppbyggda, då det blir väldigt mycket av denna vara.

Referenser

Datablad:

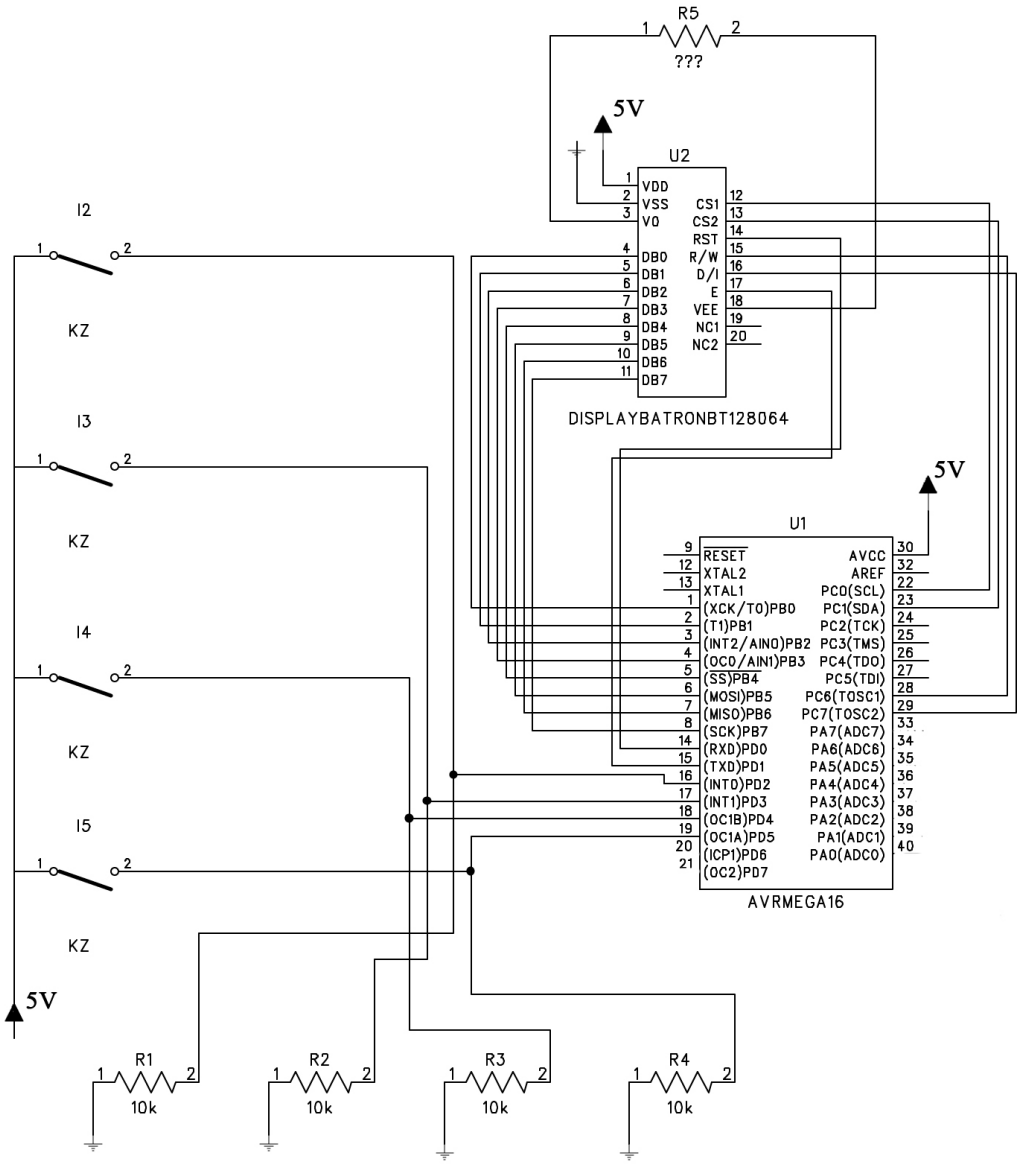
LCD-skärm GDM12864C (Red):
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/GDM12864H.pdf

Böcker:

ATMEL 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash.
Institutionen för Elektro- och informationsteknik, Lunds Tekniska Högskola, Lunds Universitet, 2012.

The C Programming Language 2th Edition, 1988
Brian W. Kernighan, Dennis M. Ritchie

Bilaga 1 – Kopplingsschema



Bilaga 2 - Källkod

Nedan följer den fullständiga källkoden för projektet uppdelad i ett antal underrubriker för lättare överskådning.

INCLUDE-STATEMENTS + KOPPLINGARNA MELLAN SKÄRM OCH PROCESSOR

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

/*
Display             Processor
DB0..DB7            PB0..PB7
RST                 PD0
E                   PD1
CS2 (vänster)       PC0
CS1 (höger)         PC1
R/W                 PC6
D/I                 PC7
*/
```

VARIBLER

```
unsigned short int highscore = 0x00;
unsigned short int score = 0x00;
unsigned short int scoreHasChanged = 0x00;
unsigned long int interrupt1 = 0;
unsigned short int gameSpeed = 100;
unsigned short int ballDirection = 2;
signed short int gameOVER = -1;
unsigned short int paddleWillMove = 0x00;
unsigned short int oldY = 0;
unsigned short int oldX = 0;

struct brick
{
        unsigned short int isActive;
        unsigned short int yStartAddress;
        unsigned short int xAddress;
        unsigned short int data;
} bricks[] = {
        0x01, 0x00, 0x00, 0x07,
        0x01, 0x00, 0x00, 0x70,
        0x01, 0x08, 0x00, 0x07,
        0x01, 0x08, 0x00, 0x70,
```

```
        0x01, 0x10, 0x00, 0x07,
        0x01, 0x10, 0x00, 0x70,
        0x01, 0x18, 0x00, 0x07,
        0x01, 0x18, 0x00, 0x70,
        0x01, 0x20, 0x00, 0x07,
        0x01, 0x20, 0x00, 0x70,
        0x01, 0x28, 0x00, 0x07,
        0x01, 0x28, 0x00, 0x70,
        0x01, 0x30, 0x00, 0x07,
        0x01, 0x30, 0x00, 0x70,
        0x01, 0x38, 0x00, 0x07,
        0x01, 0x38, 0x00, 0x70,
        0x01, 0x00, 0x01, 0x07,
        0x01, 0x00, 0x01, 0x70,
        0x01, 0x08, 0x01, 0x07,
        0x01, 0x08, 0x01, 0x70,
        0x01, 0x10, 0x01, 0x07,
        0x01, 0x10, 0x01, 0x70,
        0x01, 0x18, 0x01, 0x07,
        0x01, 0x18, 0x01, 0x70,
        0x01, 0x20, 0x01, 0x07,
        0x01, 0x20, 0x01, 0x70,
        0x01, 0x28, 0x01, 0x07,
        0x01, 0x28, 0x01, 0x70,
        0x01, 0x30, 0x01, 0x07,
        0x01, 0x30, 0x01, 0x70,
        0x01, 0x38, 0x01, 0x07,
        0x01, 0x38, 0x01, 0x70,
        0x01, 0x00, 0x02, 0x07,
        0x01, 0x00, 0x02, 0x70,
        0x01, 0x08, 0x02, 0x07,
        0x01, 0x08, 0x02, 0x70,
        0x01, 0x10, 0x02, 0x07,
        0x01, 0x10, 0x02, 0x70,
        0x01, 0x18, 0x02, 0x07,
        0x01, 0x18, 0x02, 0x70,
        0x01, 0x20, 0x02, 0x07,
        0x01, 0x20, 0x02, 0x70,
        0x01, 0x28, 0x02, 0x07,
        0x01, 0x28, 0x02, 0x70,
        0x01, 0x30, 0x02, 0x07,
        0x01, 0x30, 0x02, 0x70,
        0x01, 0x38, 0x02, 0x07,
        0x01, 0x38, 0x02, 0x70
};

struct paddle
```

```
{
	unsigned short int yStartAddress;
	unsigned short int xAddress;
	unsigned short int data;
};

struct paddle p = {	0x1A, 0x07, 0b11000000 };

struct ball
{
	unsigned short int yAddress;
	unsigned short int xAddress;
	unsigned short int data;
};

struct ball b = { 0x23, 0x07, 0b00100000 };
```

SKRIVMETODER TILL DISPLAY

```
void toggleE()
{
	PORTD |= _BV(PD1);
	PORTD &= ~_BV(PD1);
}

void clearC()
{
	PORTC &= ~_BV(PC0);
	PORTC &= ~_BV(PC1);
	PORTC &= ~_BV(PC6);
	PORTC &= ~_BV(PC7);
}

void checkStatus()
{
	unsigned short int k = 0;
	while (k < 0b11111)
	{
		k++;
	}
}

void write(unsigned short int y, unsigned short int x, unsigned short int data,
unsigned short int cs)
{
	checkStatus();
```

```
	PORTB = 0x40 + y;
	if (cs == 0x01)
		PORTC |= _BV(PC0);
	if (cs == 0x02)
		PORTC |= _BV(PC1);
	toggleE();
	clearC();

	checkStatus();
	PORTB = 0xB8 + x;
	if (cs == 0x01)
		PORTC |= _BV(PC0);
	if (cs == 0x02)
		PORTC |= _BV(PC1);
	toggleE();
	clearC();

	checkStatus();
	PORTC |= _BV(PC7);
	PORTB = data;
	if (cs == 0x01)
		PORTC |= _BV(PC0);
	if (cs == 0x02)
		PORTC |= _BV(PC1);
	toggleE();
	clearC();

	PORTB = 0x00;
}
```

BOKSTÄVER OCH SIFFROR

```
void printA(unsigned short int y, unsigned short int x)
{
	write(y, x, 0xFE, 0x02);
	y++;
	write(y, x, 0x11, 0x02);
	y++;
	write(y, x, 0x11, 0x02);
	y++;
	write(y, x, 0x11, 0x02);
	y++;
	write(y, x, 0xFE, 0x02);
}

void printB(unsigned short int y, unsigned short int x)
```

```
{
	write(y, x, 0xFF, 0x02);
	y++;
	write(y, x, 0x89, 0x02);
	y++;
	write(y, x, 0x89, 0x02);
	y++;
	write(y, x, 0x89, 0x02);
	y++;
	write(y, x, 0x76, 0x02);
}

void printC(unsigned short int y, unsigned short int x)
{
	write(y, x, 0x3C, 0x02);
	y++;
	write(y, x, 0x42, 0x02);
	y++;
	write(y, x, 0x81, 0x02);
	y++;
	write(y, x, 0x81, 0x02);
	y++;
	write(y, x, 0x42, 0x02);
}

void printD(unsigned short int y, unsigned short int x)
{
	write(y, x, 0b11111111, 0x02);
	y++;
	write(y, x, 0b10000001, 0x02);
	y++;
	write(y, x, 0b10000001, 0x02);
	y++;
	write(y, x, 0b01000010, 0x02);
	y++;
	write(y, x, 0b00111100, 0x02);
}

void printE(unsigned short int y, unsigned short int x)
{
	write(y, x, 0xFF, 0x02);
	y++;
	write(y, x, 0x89, 0x02);
	y++;
	write(y, x, 0x89, 0x02);
	y++;
	write(y, x, 0x89, 0x02);
```

```
        y++;
        write(y, x, 0x89, 0x02);
}

void printG(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x7E, 0x02);
        y++;
        write(y, x, 0x81, 0x02);
        y++;
        write(y, x, 0x91, 0x02);
        y++;
        write(y, x, 0x91, 0x02);
        y++;
        write(y, x, 0x62, 0x02);
}

void printH(unsigned short int y, unsigned short int x)
{
        write(y, x, 0xFF, 0x02);
        y++;
        write(y, x, 0x08, 0x02);
        y++;
        write(y, x, 0x08, 0x02);
        y++;
        write(y, x, 0x08, 0x02);
        y++;
        write(y, x, 0xFF, 0x02);
}

void printI(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x00, 0x02);
        y++;
        write(y, x, 0x81, 0x02);
        y++;
        write(y, x, 0xFF, 0x02);
        y++;
        write(y, x, 0x81, 0x02);
        y++;
        write(y, x, 0x00, 0x02);
}

void printK(unsigned short int y, unsigned short int x)
{
        write(y, x, 0xFF, 0x02);
        y++;
```

```
        write(y, x, 0x08, 0x02);
        y++;
        write(y, x, 0x14, 0x02);
        y++;
        write(y, x, 0x22, 0x02);
        y++;
        write(y, x, 0xC1, 0x02);
}

void printM(unsigned short int y, unsigned short int x)
{
        write(y, x, 0b11111111, 0x02);
        y++;
        write(y, x, 0b00000010, 0x02);
        y++;
        write(y, x, 0b00000100, 0x02);
        y++;
        write(y, x, 0b00000010, 0x02);
        y++;
        write(y, x, 0b11111111, 0x02);
}

void printO(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x7E, 0x02);
        y++;
        write(y, x, 0x81, 0x02);
        y++;
        write(y, x, 0x81, 0x02);
        y++;
        write(y, x, 0x81, 0x02);
        y++;
        write(y, x, 0x7E, 0x02);
}

void printR(unsigned short int y, unsigned short int x)
{
        write(y, x, 0xFF, 0x02);
        y++;
        write(y, x, 0x09, 0x02);
        y++;
        write(y, x, 0x19, 0x02);
        y++;
        write(y, x, 0x29, 0x02);
        y++;
        write(y, x, 0xC6, 0x02);
}
```

```
void printS(unsigned short int y, unsigned short int x)
{
	write(y, x, 0x46, 0x02);
	y++;
	write(y, x, 0x89, 0x02);
	y++;
	write(y, x, 0x89, 0x02);
	y++;
	write(y, x, 0x91, 0x02);
	y++;
	write(y, x, 0x62, 0x02);
}

void printT(unsigned short int y, unsigned short int x)
{
	write(y, x, 0x01, 0x02);
	y++;
	write(y, x, 0x01, 0x02);
	y++;
	write(y, x, 0xFF, 0x02);
	y++;
	write(y, x, 0x01, 0x02);
	y++;
	write(y, x, 0x01, 0x02);
}

void printU(unsigned short int y, unsigned short int x)
{
	write(y, x, 0x7F, 0x02);
	y++;
	write(y, x, 0x80, 0x02);
	y++;
	write(y, x, 0x80, 0x02);
	y++;
	write(y, x, 0x80, 0x02);
	y++;
	write(y, x, 0x7F, 0x02);
}

void printY(unsigned short int y, unsigned short int x)
{
	write(y, x, 0b00000011, 0x02);
	y++;
	write(y, x, 0b00001100, 0x02);
	y++;
	write(y, x, 0b11110000, 0x02);
```

```
        y++;
        write(y, x, 0b00001100, 0x02);
        y++;
        write(y, x, 0b00000011, 0x02);
}

void print0(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x7E, 0x02);
        y++;
        write(y, x, 0x81, 0x02);
        y++;
        write(y, x, 0x81, 0x02);
        y++;
        write(y, x, 0x81, 0x02);
        y++;
        write(y, x, 0x7E, 0x02);
}

void print1(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x00, 0x02);
        y++;
        write(y, x, 0x82, 0x02);
        y++;
        write(y, x, 0xFF, 0x02);
        y++;
        write(y, x, 0x80, 0x02);
        y++;
        write(y, x, 0x00, 0x02);
}

void print2(unsigned short int y, unsigned short int x)
{
        write(y, x, 0xC2, 0x02);
        y++;
        write(y, x, 0xA1, 0x02);
        y++;
        write(y, x, 0x91, 0x02);
        y++;
        write(y, x, 0x89, 0x02);
        y++;
        write(y, x, 0x86, 0x02);
}

void print3(unsigned short int y, unsigned short int x)
{
```

```
        write(y, x, 0x42, 0x02);
        y++;
        write(y, x, 0x81, 0x02);
        y++;
        write(y, x, 0x89, 0x02);
        y++;
        write(y, x, 0x95, 0x02);
        y++;
        write(y, x, 0x62, 0x02);
}

void print4(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x38, 0x02);
        y++;
        write(y, x, 0x24, 0x02);
        y++;
        write(y, x, 0x22, 0x02);
        y++;
        write(y, x, 0xFF, 0x02);
        y++;
        write(y, x, 0x20, 0x02);
}

void print5(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x4F, 0x02);
        y++;
        write(y, x, 0x91, 0x02);
        y++;
        write(y, x, 0x91, 0x02);
        y++;
        write(y, x, 0x91, 0x02);
        y++;
        write(y, x, 0x61, 0x02);
}

void print6(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x78, 0x02);
        y++;
        write(y, x, 0x94, 0x02);
        y++;
        write(y, x, 0x92, 0x02);
        y++;
        write(y, x, 0x91, 0x02);
        y++;
```

```
        write(y, x, 0x61, 0x02);
}

void print7(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x03, 0x02);
        y++;
        write(y, x, 0xC1, 0x02);
        y++;
        write(y, x, 0x31, 0x02);
        y++;
        write(y, x, 0x0D, 0x02);
        y++;
        write(y, x, 0x03, 0x02);
}

void print8(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x76, 0x02);
        y++;
        write(y, x, 0x89, 0x02);
        y++;
        write(y, x, 0x89, 0x02);
        y++;
        write(y, x, 0x89, 0x02);
        y++;
        write(y, x, 0x76, 0x02);
}

void print9(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x86, 0x02);
        y++;
        write(y, x, 0x89, 0x02);
        y++;
        write(y, x, 0x49, 0x02);
        y++;
        write(y, x, 0x29, 0x02);
        y++;
        write(y, x, 0x1E, 0x02);
}

void printQM(unsigned short int y, unsigned short int x)
{
        write(y, x, 0x02, 0x02);
        y++;
        write(y, x, 0x01, 0x02);
```

```
        y++;
        write(y, x, 0xB1, 0x02);
        y++;
        write(y, x, 0x09, 0x02);
        y++;
        write(y, x, 0x06, 0x02);
}
```

UTSKRIFTER

```
void printBrickBuster()
{
        printB(0x04, 0x00);
        printR(0x0A, 0x00);
        printI(0x10, 0x00);
        printC(0x16, 0x00);
        printK(0x1C, 0x00);

        printB(0x10, 0x01);
        printU(0x16, 0x01);
        printS(0x1C, 0x01);
        printT(0x22, 0x01);
        printE(0x28, 0x01);
        printR(0x2E, 0x01);
}

void printHighScore()
{
        printH(0x04, 0x03);
        printI(0x0A, 0x03);
        printG(0x10, 0x03);
        printH(0x16, 0x03);
        printS(0x1C, 0x03);
        printC(0x22, 0x03);
        printO(0x28, 0x03);
        printR(0x2E, 0x03);
        printE(0x34, 0x03);
}

void printYourScore()
{
        printY(0x04, 0x06);
        printO(0x0A, 0x06);
        printU(0x10, 0x06);
        printR(0x16, 0x06);
        printS(0x22, 0x06);
```

```
	printC(0x28, 0x06);
	printO(0x2E, 0x06);
	printR(0x34, 0x06);
	printE(0x3A, 0x06);
}

void printStart()
{
	printS(0x10, 0x06);
	printT(0x16, 0x06);
	printA(0x1C, 0x06);
	printR(0x22, 0x06);
	printT(0x28, 0x06);
	printQM(0x2E, 0x06);
}

void printChooseMode()
{
	printC(0x04, 0x03);
	printH(0x0A, 0x03);
	printO(0x10, 0x03);
	printO(0x16, 0x03);
	printS(0x1C, 0x03);
	printE(0x22, 0x03);

	printM(0x16, 0x04);
	printO(0x1C, 0x04);
	printD(0x22, 0x04);
	printE(0x28, 0x04);
}

void printScore(unsigned short int whichScore, unsigned short int xAddress)
{
	switch (whichScore / 10)
	{
		case 1:
			print1(0x16, xAddress);
			break;
		case 2:
			print2(0x16, xAddress);
			break;
		case 3:
			print3(0x16, xAddress);
			break;
		case 4:
			print4(0x16, xAddress);
			break;
```

```
	}

	switch (whichScore % 10)
	{
		case 1:
			print1(0x1C, xAddress);
			break;
		case 2:
			print2(0x1C, xAddress);
			break;
		case 3:
			print3(0x1C, xAddress);
			break;
		case 4:
			print4(0x1C, xAddress);
			break;
		case 5:
			print5(0x1C, xAddress);
			break;
		case 6:
			print6(0x1C, xAddress);
			break;
		case 7:
			print7(0x1C, xAddress);
			break;
		case 8:
			print8(0x1C, xAddress);
			break;
		case 9:
			print9(0x1C, xAddress);
			break;
		case 0:
			print0(0x1C, xAddress);
			break;
	}
}
```

HANTERING AV BLOCKEN

```
void paintBricks(unsigned short int start, unsigned short int stop)
{
	for (start; start < stop; start += 2)
	{
		if (bricks[start].isActive == 0x01)
		{
			if (bricks[start+1].isActive == 0x01)
```

```
                        {
                                for (unsigned short int k = 0x00; k < 0x07; k++)
                                {
                                        write(bricks[start].yStartAddress + k,
bricks[start].xAddress, bricks[start].data + bricks[start+0x01].data, 0x01);
                                }
                        }
                        else
                        {
                                for (unsigned short int k = 0x00; k < 0x07; k++)
                                {
                                        write(bricks[start].yStartAddress + k,
bricks[start].xAddress, bricks[start].data, 0x01);
                                }
                        }
                }
                else if (bricks[start+1].isActive == 0x01)
                {
                        for (unsigned short int k = 0x00; k < 0x07; k++)
                        {
                                write(bricks[start].yStartAddress + k,
bricks[start].xAddress, bricks[start+0x01].data, 0x01);
                        }
                }
        }
}

void paintWhichBricks(unsigned short int yAddress, unsigned short int xAddress)
{
        switch (xAddress)
        {
                case 0x00:
                        if (yAddress < 0x07)
                        {
                                paintBricks(0, 1);
                        }
                        else if (yAddress >= 0x08 && yAddress < 0x0F)
                        {
                                paintBricks(2, 3);
                        }
                        else if (yAddress >= 0x10 && yAddress < 0x18)
                        {
                                paintBricks(4, 5);
                        }
                        else if (yAddress >= 0x18 && yAddress < 0x1F)
                        {
                                paintBricks(6, 7);
```

```
		}
		else if (yAddress >= 0x20 && yAddress < 0x28)
		{
			paintBricks(8, 9);
		}
		else if (yAddress >= 0x28 && yAddress < 0x2F)
		{
			paintBricks(10, 11);
		}
		else if (yAddress >= 0x30 && yAddress < 0x38)
		{
			paintBricks(12, 13);
		}
		else if (yAddress >= 0x38 && yAddress < 0x3F)
		{
			paintBricks(14, 15);
		}
		break;
	case 0x01:
		if (yAddress < 0x07)
			paintBricks(16, 17);
		else if (yAddress >= 0x08 && yAddress < 0x0F)
			paintBricks(18, 19);
		else if (yAddress >= 0x10 && yAddress < 0x18)
			paintBricks(20, 21);
		else if (yAddress >= 0x18 && yAddress < 0x1F)
			paintBricks(22, 23);
		else if (yAddress >= 0x20 && yAddress < 0x28)
			paintBricks(24, 25);
		else if (yAddress >= 0x28 && yAddress < 0x2F)
			paintBricks(26, 27);
		else if (yAddress >= 0x30 && yAddress < 0x38)
			paintBricks(28, 29);
		else if (yAddress >= 0x38 && yAddress < 0x3F)
			paintBricks(30, 31);
		break;
	case 0x02:
		if (yAddress < 0x07)
			paintBricks(32, 33);
		else if (yAddress >= 0x08 && yAddress < 0x0F)
			paintBricks(34, 35);
		else if (yAddress >= 0x10 && yAddress < 0x18)
			paintBricks(36, 37);
		else if (yAddress >= 0x18 && yAddress < 0x1F)
			paintBricks(38, 39);
		else if (yAddress >= 0x20 && yAddress < 0x28)
			paintBricks(40, 41);
```

```
                else if (yAddress >= 0x28 && yAddress < 0x2F)
                        paintBricks(42, 43);
                else if (yAddress >= 0x30 && yAddress < 0x38)
                        paintBricks(44, 45);
                else if (yAddress >= 0x38 && yAddress < 0x3F)
                        paintBricks(46, 47);
                break;
        }
}
```

HANTERING AV PADDELN

```
void movePaddleLeft()
{
        if (p.yStartAddress > 0x00)
        {
                p.yStartAddress--;
                paintPaddle();
                write(p.yStartAddress + 0x0C, p.xAddress, 0x00, 0x01);
        }
}

void movePaddleRight()
{
        if (p.yStartAddress < 0x33)
        {
                p.yStartAddress++;
                paintPaddle();
                write(p.yStartAddress - 0x01, p.xAddress, 0x00, 0x01);
        }
}

void paintPaddle()
{
        for (unsigned short int k = 0x00; k < 0x0C; k++)
        {
                write(p.yStartAddress + k, p.xAddress, p.data, 0x01);
        }
}
```

HANTERING AV BOLLEN

```
unsigned short int hitBrick(unsigned short int i)
{
```

```
	if (bricks[i].isActive == 0x01)
	{
		bricks[i].isActive = 0x00;
		score++;
		scoreHasChanged = 0x01;
		if (i % 2 == 0)
		{
			paintBricks(i, i + 1);
		}
		else
		{
			paintBricks(i - 1, i);
		}
		unsigned short int data = b.data;
		unsigned short int y = b.yAddress;
		unsigned short int where;
		switch (ballDirection)
		{
			case 1:
				if (data == 0x40 || data == 0x04)
				{
					if (y == 0x06 || y == 0x0E || y == 0x16 || y == 
0x1E || y == 0x26 || y == 0x2E || y == 0x36 || y == 0x3E)
					{
						where = 5;
					}
					else
					{
						where = 6;
					}
				}
				else if (data == 0x20 || data == 0x10 || data == 0x02 || 
data == 0x01)
				{
					where = 4;
				}
				break;
			case 2:
				where = 6;
				break;
			case 3:
				if (data == 0x40 || data == 0x04)
				{
					if (y == 0x00 || y == 0x08 || y == 0x10 || y == 
0x18 || y == 0x20 || y == 0x28 || y == 0x30 || y == 0x38)
					{
						where = 7;
```

```
                        }
                        else
                        {
                                where = 6;
                        }
                }
                else if (data == 0x20 || data == 0x10 || data == 0x02 ||
data == 0x01)
                {
                        where = 8;
                }
                break;
        case 4:
                if (data == 0x10 || data == 0x01)
                {
                        if (y == 0x00 || y == 0x08 || y == 0x10 || y ==
0x18 || y == 0x20 || y == 0x28 || y == 0x30 || y == 0x38)
                        {
                                where = 1;
                        }
                        else
                        {
                                where = 2;
                        }
                }
                else if (data == 0x20 || data == 0x40 || data == 0x02 ||
data == 0x04)
                {
                        where = 8;
                }
                break;
        case 5:
                where = 2;
                break;
        case 6:
                if (data == 0x10 || data == 0x01)
                {
                        if (y == 0x06 || y == 0x0E || y == 0x16 || y ==
0x1E || y == 0x26 || y == 0x2E || y == 0x36 || y == 0x3E)
                        {
                                where = 3;
                        }
                        else
                        {
                                where = 2;
                        }
                }
```

```
                else if (data == 0x20 || data == 0x40 || data == 0x02 ||
data == 0x04)
                {
                        where = 4;
                }
                break;
        }

        switch (ballDirection)
        {
                case 1:
                        switch (where)
                        {
                                case 4:
                                        ballDirection = 3;
                                        break;
                                case 5:
                                        ballDirection = 4;
                                        break;
                                case 6:
                                        ballDirection = 6;
                                        break;
                        }
                        break;
                case 2:
                        switch (where)
                        {
                                case 6:
                                        ballDirection = 5;
                                        break;
                        }
                        break;
                case 3:
                        switch (where)
                        {
                                case 6:
                                        ballDirection = 4;
                                        break;
                                case 7:
                                        ballDirection = 6;
                                        break;
                                case 8:
                                        ballDirection = 1;
                                        break;
                        }
                        break;
                case 4:
```

```
                switch (where)
                {
                        case 1:
                                ballDirection = 1;
                                break;
                        case 2:
                                ballDirection = 3;
                                break;
                        case 8:
                                ballDirection = 6;
                                break;
                }
                break;
        case 5:
                switch (where)
                {
                        case 2:
                                ballDirection = 2;
                                break;
                }
                break;
        case 6:
                switch (where)
                {
                        case 2:
                                ballDirection = 1;
                                break;
                        case 3:
                                ballDirection = 3;
                                break;
                        case 4:
                                ballDirection = 4;
                                break;
                }
                break;
}
switch (ballDirection)
{
        case 1:
                moveBall(-1, -1);
                break;
        case 2:
                moveBall(0, -1);
                break;
        case 3:
                moveBall(1, -1);
                break;
```

```
                        case 4:
                                moveBall(1, 1);
                                break;
                        case 5:
                                moveBall(0, 1);
                                break;
                        case 6:
                                moveBall(-1, 1);
                                break;
                }
                paintPaddle();
                paintBall();
                return 1;
        }
        return 0;
}

unsigned short int willHitBrick()
{
        unsigned short int isHit = 0;
        unsigned short int y = b.yAddress;
        if (b.xAddress == 0x02)
        {
                if (b.data <= 0x40 && b.data >= 0x10)
                {
                        if (y >= 0x00 && y <= 0x06)
                        {
                                isHit = hitBrick(33);
                        }
                        else if (y >= 0x08 && y <= 0x0E)
                        {
                                isHit = hitBrick(35);
                        }
                        else if (y >= 0x10 && y <= 0x16)
                        {
                                isHit = hitBrick(37);
                        }
                        else if (y >= 0x18 && y <= 0x1E)
                        {
                                isHit = hitBrick(39);
                        }
                        else if (y >= 0x20 && y <= 0x26)
                        {
                                isHit = hitBrick(41);
                        }
                        else if (y >= 0x28 && y <= 0x2E)
                        {
```

```
                        isHit = hitBrick(43);
                }
                else if (y >= 0x30 && y <= 0x36)
                {
                        isHit = hitBrick(45);
                }
                else if (y >= 0x38 && y <= 0x3E)
                {
                        isHit = hitBrick(47);
                }
        }
        else if (b.data <= 0x04 && b.data >= 0x01)
        {
                if (y >= 0x00 && y <= 0x06)
                {
                        isHit = hitBrick(32);
                }
                else if (y >= 0x08 && y <= 0x0E)
                {
                        isHit = hitBrick(34);
                }
                else if (y >= 0x10 && y <= 0x16)
                {
                        isHit = hitBrick(36);
                }
                else if (y >= 0x18 && y <= 0x1E)
                {
                        isHit = hitBrick(38);
                }
                else if (y >= 0x20 && y <= 0x26)
                {
                        isHit = hitBrick(40);
                }
                else if (y >= 0x28 && y <= 0x2E)
                {
                        isHit = hitBrick(42);
                }
                else if (y >= 0x30 && y <= 0x36)
                {
                        isHit = hitBrick(44);
                }
                else if (y >= 0x38 && y <= 0x3E)
                {
                        isHit = hitBrick(46);
                }
        }
}
```

```
else if (b.xAddress == 0x01)
{
	if (b.data <= 0x40 && b.data >= 0x10)
	{
		if (y >= 0x00 && y <= 0x06)
		{
			isHit = hitBrick(17);
		}
		else if (y >= 0x08 && y <= 0x0E)
		{
			isHit = hitBrick(19);
		}
		else if (y >= 0x10 && y <= 0x16)
		{
			isHit = hitBrick(21);
		}
		else if (y >= 0x18 && y <= 0x1E)
		{
			isHit = hitBrick(23);
		}
		else if (y >= 0x20 && y <= 0x26)
		{
			isHit = hitBrick(25);
		}
		else if (y >= 0x28 && y <= 0x2E)
		{
			isHit = hitBrick(27);
		}
		else if (y >= 0x30 && y <= 0x36)
		{
			isHit = hitBrick(29);
		}
		else if (y >= 0x38 && y <= 0x3E)
		{
			isHit = hitBrick(31);
		}
	}
	else if (b.data <= 0x04 && b.data >= 0x01)
	{
		if (y >= 0x00 && y <= 0x06)
		{
			isHit = hitBrick(16);
		}
		else if (y >= 0x08 && y <= 0x0E)
		{
			isHit = hitBrick(18);
		}
```

```
                else if (y >= 0x10 && y <= 0x16)
                {
                        isHit = hitBrick(20);
                }
                else if (y >= 0x18 && y <= 0x1E)
                {
                        isHit = hitBrick(22);
                }
                else if (y >= 0x20 && y <= 0x26)
                {
                        isHit = hitBrick(24);
                }
                else if (y >= 0x28 && y <= 0x2E)
                {
                        isHit = hitBrick(26);
                }
                else if (y >= 0x30 && y <= 0x36)
                {
                        isHit = hitBrick(28);
                }
                else if (y >= 0x38 && y <= 0x3E)
                {
                        isHit = hitBrick(30);
                }
        }
}
else if (b.xAddress == 0x00)
{
        if (b.data <= 0x40 && b.data >= 0x10)
        {
                if (y >= 0x00 && y <= 0x06)
                {
                        isHit = hitBrick(1);
                }
                else if (y >= 0x08 && y <= 0x0E)
                {
                        isHit = hitBrick(3);
                }
                else if (y >= 0x10 && y <= 0x16)
                {
                        isHit = hitBrick(5);
                }
                else if (y >= 0x18 && y <= 0x1E)
                {
                        isHit = hitBrick(7);
                }
                else if (y >= 0x20 && y <= 0x26)
```

```
		{
			isHit = hitBrick(9);
		}
		else if (y >= 0x28 && y <= 0x2E)
		{
			isHit = hitBrick(11);
		}
		else if (y >= 0x30 && y <= 0x36)
		{
			isHit = hitBrick(13);
		}
		else if (y >= 0x38 && y <= 0x3E)
		{
			isHit = hitBrick(15);
		}
	}
	else if (b.data <= 0x04 && b.data >= 0x01)
	{
		if (y >= 0x00 && y <= 0x06)
		{
			isHit = hitBrick(0);
		}
		else if (y >= 0x08 && y <= 0x0E)
		{
			isHit = hitBrick(2);
		}
		else if (y >= 0x10 && y <= 0x16)
		{
			isHit = hitBrick(4);
		}
		else if (y >= 0x18 && y <= 0x1E)
		{
			isHit = hitBrick(6);
		}
		else if (y >= 0x20 && y <= 0x26)
		{
			isHit = hitBrick(8);
		}
		else if (y >= 0x28 && y <= 0x2E)
		{
			isHit = hitBrick(10);
		}
		else if (y >= 0x30 && y <= 0x36)
		{
			isHit = hitBrick(12);
		}
		else if (y >= 0x38 && y <= 0x3E)
```

```
                        {
                                isHit = hitBrick(14);
                        }
                }
        }
        return isHit;
}

void moveBall(signed short int y, signed short int x)
{
        oldY = b.yAddress;
        oldX = b.xAddress;

        switch (b.data)
        {
                case 0b00000001:
                        if (b.xAddress == 0x00 &&  x < 0)
                        {
                                x = 1;
                        }
                        if (x < 0)
                        {
                                b.xAddress--;
                                b.data = 0b10000000;
                        }
                        if (x > 0)
                        {
                                b.data = 0b00000010;
                        }
                        break;
                case 0b00000010:
                        if (x < 0)
                        {
                                b.data = 0b00000001;
                        }
                        if (x > 0)
                        {
                                b.data = 0b00000100;
                        }
                        break;
                case 0b00000100:
                        if (x < 0)
                        {
                                b.data = 0b00000010;
                        }
                        if (x > 0)
                        {
```

```
                        b.data = 0b00001000;
                }
                break;
        case 0b00001000:
                if (x < 0)
                {
                        b.data = 0b00000100;
                }
                if (x > 0)
                {
                        b.data = 0b00010000;
                }
                break;
        case 0b00010000:
                if (x < 0)
                {
                        b.data = 0b00001000;
                }
                if (x > 0)
                {
                        b.data = 0b00100000;
                }
                break;
        case 0b00100000:
                if (b.xAddress == 0x07 && x > 0)
                {
                        if (p.yStartAddress >= b.yAddress - 0x0B &&
p.yStartAddress <= b.yAddress - 0x08)
                        {
                                x = -1;
                                y = 1;
                        }
                        if (p.yStartAddress + 0x07 >= b.yAddress &&
p.yStartAddress + 0x04 <= b.yAddress)
                        {
                                x = -1;
                                y = 0;
                        }
                        if (p.yStartAddress + 0x03 >= b.yAddress &&
p.yStartAddress <= b.yAddress)
                        {
                                x = -1;
                                y = -1;
                        }
                        if (p.yStartAddress + 0x0B < b.yAddress ||
p.yStartAddress > b.yAddress)
                        {
```

```
                                        gameOver();
                                }
                        }
                        if (x < 0)
                        {
                                b.data = 0b00010000;
                        }
                        if (x > 0)
                        {
                                b.data = 0b01000000;
                        }
                        break;
                case 0b01000000:
                        if (x < 0)
                        {
                                b.data = 0b00100000;
                        }
                        if (x > 0)
                        {
                                b.data = 0b10000000;
                        }
                        break;
                case 0b10000000:
                        if (x < 0)
                        {
                                b.data = 0b01000000;
                        }
                        if (x > 0)
                        {
                                b.xAddress++;
                                b.data = 0b00000001;
                                if (b.xAddress == 0x03)
                                {
                                        paintBricks(0x00, 0x30);
                                }
                        }
                        break;
        }

        if (b.yAddress + y < 0x00 || b.yAddress + y > 0x3F)
        {
                y *= -1;
        }
        b.yAddress += y;

        if (willHitBrick() != 1)
        {
```

```
                if (x == -1 && y == -1)
                {
                        ballDirection = 1;
                }
                else if (x == -1 && y == 0)
                {
                        ballDirection = 2;
                }
                else if (x == -1 && y == 1)
                {
                        ballDirection = 3;
                }
                else if (x == 1 && y == 1)
                {
                        ballDirection = 4;
                }
                else if (x == 1 && y == 0)
                {
                        ballDirection = 5;
                }
                else if (x == 1 && y == -1)
                {
                        ballDirection = 6;
                }
                write(oldY, oldX, 0x00, 0x01);
                paintWhichBricks(oldY, oldX);
        }
        paintBricks(0x00, 0x30);
}

void addBricks(unsigned short int first, unsigned short int second)
{
        if (bricks[first].isActive == 0x01 && bricks[second].isActive == 0x01)
        {
                write(b.yAddress, b.xAddress, b.data + 0b01110111, 0x01);
        }
        else if (bricks[first].isActive == 0x01)
        {
                write(b.yAddress, b.xAddress, b.data + 0b00000111, 0x01);
        }
        else if (bricks[second].isActive == 0x01)
        {
                write(b.yAddress, b.xAddress, b.data + 0b01110000, 0x01);
        }
}

void addWhichBricks()
```

```
{
	switch (b.xAddress)
	{
		case 0x00:
			if (b.yAddress < 0x07)
				addBricks(0, 1);
			else if (b.yAddress >= 0x08 && b.yAddress < 0x0F)
				addBricks(2, 3);
			else if (b.yAddress >= 0x10 && b.yAddress < 0x18)
				addBricks(4, 5);
			else if (b.yAddress >= 0x18 && b.yAddress < 0x1F)
				addBricks(6, 7);
			else if (b.yAddress >= 0x20 && b.yAddress < 0x28)
				addBricks(8, 9);
			else if (b.yAddress >= 0x28 && b.yAddress < 0x2F)
				addBricks(10, 11);
			else if (b.yAddress >= 0x30 && b.yAddress < 0x38)
				addBricks(12, 13);
			else if (b.yAddress >= 0x38 && b.yAddress < 0x3F)
				addBricks(14, 15);
			break;
		case 0x01:
			if (b.yAddress < 0x07)
				addBricks(16, 17);
			else if (b.yAddress >= 0x08 && b.yAddress < 0x0F)
				addBricks(18, 19);
			else if (b.yAddress >= 0x10 && b.yAddress < 0x18)
				addBricks(20, 21);
			else if (b.yAddress >= 0x18 && b.yAddress < 0x1F)
				addBricks(22, 23);
			else if (b.yAddress >= 0x20 && b.yAddress < 0x28)
				addBricks(24, 25);
			else if (b.yAddress >= 0x28 && b.yAddress < 0x2F)
				addBricks(26, 27);
			else if (b.yAddress >= 0x30 && b.yAddress < 0x38)
				addBricks(28, 29);
			else if (b.yAddress >= 0x38 && b.yAddress < 0x3F)
				addBricks(30, 31);
			break;
		case 0x02:
			if (b.yAddress < 0x07)
				addBricks(32, 33);
			else if (b.yAddress >= 0x08 && b.yAddress < 0x0F)
				addBricks(34, 35);
			else if (b.yAddress >= 0x10 && b.yAddress < 0x18)
				addBricks(36, 37);
			else if (b.yAddress >= 0x18 && b.yAddress < 0x1F)
```

```
                addBricks(38, 39);
            else if (b.yAddress >= 0x20 && b.yAddress < 0x28)
                addBricks(40, 41);
            else if (b.yAddress >= 0x28 && b.yAddress < 0x2F)
                addBricks(42, 43);
            else if (b.yAddress >= 0x30 && b.yAddress < 0x38)
                addBricks(44, 45);
            else if (b.yAddress >= 0x38 && b.yAddress < 0x3F)
                addBricks(46, 47);
            break;
        }
}

void paintBall()
{
        if (b.xAddress == 0x07)
        {
                if (p.yStartAddress >= b.yAddress - 0x0B && p.yStartAddress <=
b.yAddress)
                {
                        write(b.yAddress, b.xAddress, b.data + p.data, 0x01);
                }
        }
        else if (b.xAddress > 0x02)
        {
                write(b.yAddress, b.xAddress, b.data, 0x01);
        }
        else
        {
                addWhichBricks();
        }
}
```

UPPSTARTS- OCH ÅTERSTÄLLNINGSMETODER

```
void initialize()
{
        DDRD = 0xc3;
        DDRC |= _BV(PC0);
        DDRC |= _BV(PC1);
        DDRC |= _BV(PC6);
        DDRC |= _BV(PC7);
        DDRB = 0xFF;
        DDRA = 0xFF;

        PORTD |= _BV(PD0);
```

```
	PORTD &= ~_BV(PD1);
	PORTD |= _BV(PD7);
	PORTD |= _BV(PD6);
	clearC();
	PORTB = 0x00;
	PORTA = 0x00;

	TCCR1B = 0b00000010;
	TIMSK = 0b00000100;

	MCUCR = 0b00001111;
	GICR = 0b11000000;
}

void toggleDisplay(unsigned short int onOff, unsigned short int cs)
{
	if (cs == 0x01)
	{
		PORTC |= _BV(PC0);
	}
	if (cs == 0x02)
	{
		PORTC |= _BV(PC1);
	}
	PORTB = 0x3E + onOff;
	toggleE();
	clearC();
}

void clearDisplay(unsigned short int cs)
{
	for (unsigned short int x=0; x<8; x++)
	{
		for (unsigned short int y=0; y<64; y++)
		{
			write(y, x, 0x00, cs);
		}
	}
}

void printMenu()
{
	printBrickBuster();
	printHighScore();
	printScore(highscore, 0x04);
	printYourScore();
	printScore(score, 0x07);
```

```
}

void resetPlayground()
{
	p.yStartAddress = 0x1A;
	p.xAddress = 0x07;
	p.data = 0xC0;
	paintPaddle();

	b.yAddress = 0x23;
	b.xAddress = 0x07;
	b.data = 0x20;
	paintBall();
	ballDirection = 0x02;
}

void gameOver()
{
	gameOVER = 1;
}
```

AVBROTT

```
unsigned long int counter = 0;

ISR(TIMER1_OVF_vect)
{
	if (gameOVER < 0)
	{
		if (counter % 2 == 0)
		{
			switch (ballDirection)
			{
				case 1:
					moveBall(-1, -1);
					break;
				case 2:
					moveBall(0, -1);
					break;
				case 3:
					moveBall(1, -1);
					break;
				case 4:
					moveBall(1, 1);
					break;
				case 5:
```

```
                                        moveBall(0, 1);
                                        break;
                                case 6:
                                        moveBall(-1, 1);
                                        break;
                        }
                        if (counter % 18 == 0)
                        {
                                for (unsigned short int x=3; x<8; x++)
                                {
                                        for (unsigned short int y=0; y<64; y++)
                                        {
                                                write(y, x, 0x00, 0x01);
                                        }
                                }
                        }
                        if (counter % 120 == 0)
                        {
                                clearDisplay(0x01);
                                paintBricks(0x00, 0x30);
                        }
                        if (counter % 120 == 0)
                        {
                                clearDisplay(0x02);
                                printMenu();
                        }
                        paintPaddle();
                        paintBall();
                }
                counter++;
        }
}

ISR(INT0_vect)
{
        if (gameOVER < 0)
        {
                if (paddleWillMove > 0x01)
                {
                        movePaddleLeft();
                        paddleWillMove = 0;
                }
                paddleWillMove++;
        }
        else
        {
                startGame();
```

```
	}
}

ISR(INT1_vect)
{
	if (gameOVER < 0)
	{
		if (paddleWillMove > 0x01)
		{
			movePaddleRight();
			paddleWillMove = 0;
		}
		paddleWillMove++;
	}
	else
	{
		startGame();
	}
}
```

HUVUDPROGRAM

```
void mode1()
{
	for (int k = 0; k < 0x30; k++)
	{
		bricks[k].isActive = 0x01;
	}
}

void mode2()
{
	bricks[0].isActive = 0x00;
	bricks[1].isActive = 0x00;
	bricks[2].isActive = 0x00;
	bricks[12].isActive = 0x00;
	bricks[14].isActive = 0x00;
	bricks[15].isActive = 0x00;
	bricks[32].isActive = 0x00;
	bricks[33].isActive = 0x00;
	bricks[35].isActive = 0x00;
	bricks[45].isActive = 0x00;
	bricks[46].isActive = 0x00;
	bricks[47].isActive = 0x00;
}
```

```
void pause()
{
	cli();
	unsigned short int pinD = PIND & 0b00100000;
	while (pinD == 0b00000000)
	{
		pinD = PIND & 0b00100000;
	}
	sei();
}

void mainLoop()
{
	sei();
	unsigned short int pinD = PIND & 0b00010000;
	while(1)
	{
		pinD = PIND & 0b00010000;
		if (pinD == 0b00010000)
		{
			pause();
		}
		if (scoreHasChanged == 0x01)
		{
			printScore(score, 0x07);
			scoreHasChanged == 0x00;
		}
		if (gameOVER > 0)
		{
			cli();
			break;
		}
	}
	printScore(score, 0x07);
	highscore = score;
	score = 0x00;
	chooseMode();
}

void startGame()
{
	clearDisplay(0x02);
	printBrickBuster();
	printHighScore();
	printScore(highscore, 0x04);
	printYourScore();
	printScore(score, 0x07);
```

```
	gameOVER = -1;
	mainLoop();
}

void chooseMode()
{
	clearDisplay(0x02);
	printBrickBuster();
	printChooseMode();
	printStart();
	unsigned short int pinD = PIND & 0b00110000;
	while (pinD == 0b00000000)
	{
		pinD = PIND & 0b00110000;
		if (pinD == 0b00010000)
		{
			clearDisplay(0x01);
			mode1();
			paintBricks(0x00, 0x30);
		}
		if (pinD == 0b00100000)
		{
			clearDisplay(0x01);
			mode1();
			mode2();
			paintBricks(0x00, 0x30);
		}
	}
	resetPlayground();
	startGame();
}

void main(void)
{
	initialize();
	toggleDisplay(0x01, 0x01);
	toggleDisplay(0x01, 0x02);
	clearDisplay(0x01);
	clearDisplay(0x02);
	chooseMode();
}
```