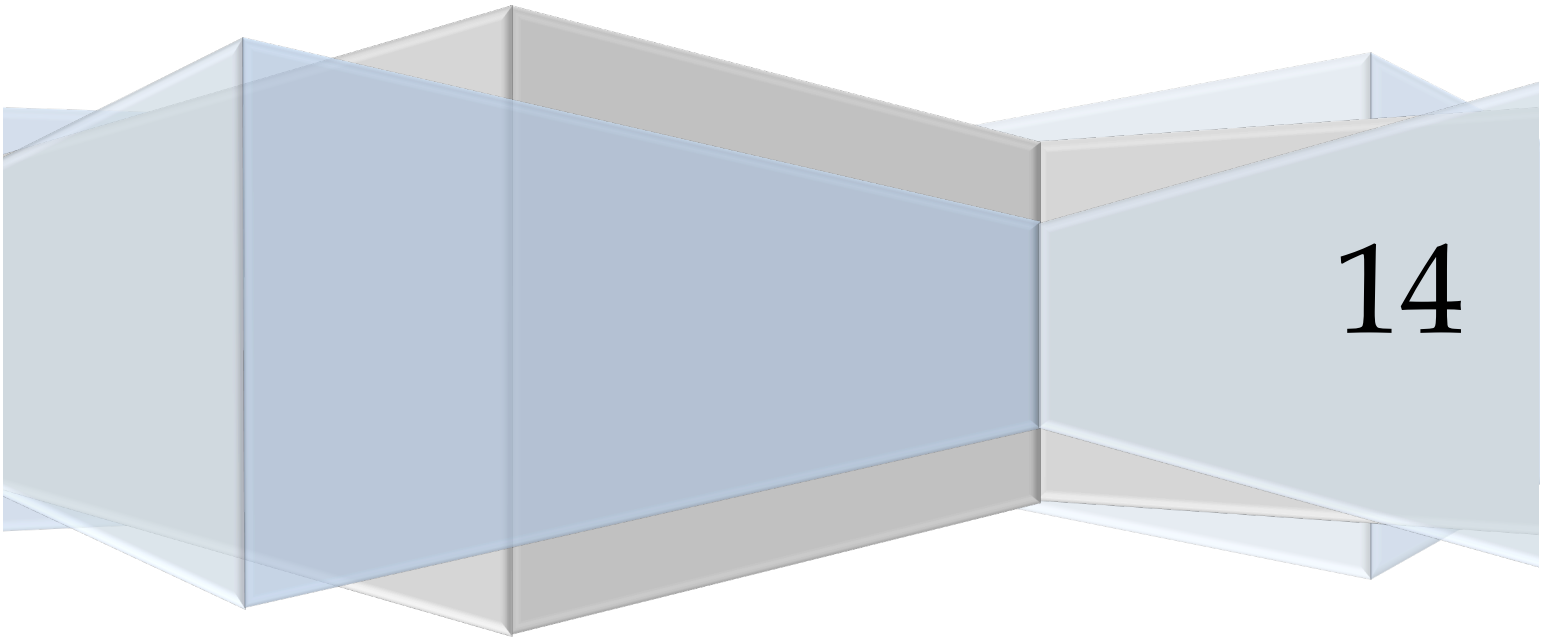
**EITF11 – Digitala Projekt**
**Handledare: Bertil Lindvall**

# Flappy Bird

## Spelkonsol och spel

**Oskar Johansson, Richard Luong och Johan Mattisson**



# Inledning

Den här rapporten är en del av kursen EITF11, Digitala projekt som ges bland annat till studenter på Industriell ekonomi vid LTH. Kursen erbjuder studenter möjligheten att i grupp driva ett projekt som är mer tekniskt intensivt än något av de projekt som tidigare genomförts på utbildningen.

Just detta projekt ska resultera i en spelkonsoll på vilken man ska kunna spela ett spel vid namn Flappy Bird. Flappy Bird är ett spel ursprungligen utvecklat av Dong Nguyen, men som han tog bort från App Store. (Wikipedia, 2014a)

Insignaler till spelet ges av en enkel knapp. Den här knappen startar inte bara spelet, utan det är även den man trycker på för att få fågeln att flyga uppåt. Med samma knapp kan man även starta om spelet när fågeln dör.

Utsignaler är förstås LCD-displayen. För att spelaren ska få en upplevelse hon sent kommer att glömma är även en högtalare inkopplad till spelkonsolen.

# Kravspecifikation

- Det ska finnas en högtalare som spelar upp ljud.
- Det finns endast en knapp. Denna används både för att spela spelat samt navigera i menyn.
- Spelet startas genom att knappen trycks ner
- Fågeln flaxar (förflyttas upp i y-led) varje gång knappen trycks ner, samtidigt som rör tillkommer från höger sida
- Då tiden går utan att knappen trycks ned så ska fågeln falla nedåt
- Krockar fågeln med ett rör, så dör den
- Faller fågeln till marken eller flyger upp i taket, så dör den
- Det ska finnas en skärm med tilltalande användargränssnit
- Spelet ska kunna räkna poäng. Varje gång fågeln passerar igenom ett rör utan att krocka så ska poängen öka.
- Spelet ska kunna visa resultat efter avslutad spelomgång.

# Hårdvara

För fullständigt kopplingschema, se bilaga.

## Mikrokontroller

Som mikrokontroller användes en ATMega32, en Atmel®AVR® 8-bitars mikrokontroller. Mikrokontrollern innehåller 32kb minne, dubbelt så mycket som den ursprungliga mikrokontrollern som initials användes, ATMega16. Det som gjorde att det större minnet behövde användas var för att kunna representera alla skärmens pixlar i en matris. Mikrokontrollern har 32 programmeringsbara I/O pinnar fördelade över 4 portar: PORTA-PORTD.

### PORTA
Användes för att ta emot information från knappen.

### PORTB
Användes för att skicka instruktioner från processorn till skärmen. Dessa instruktioner påverkar hur skärmen behandlar de databitar som sedan skickas på PORTD. Användes även för att skicka analog data till högtalaren.

### PORTC
Användes för debuggning med JTAG. Det är via JTAGen vi kunde styra och programmera processorn från datorn.

### PORTD
Används för att att skicka databitar till skärmen.

# Display - GDM12864H 128 x 64.

Som display användes en LCD-display med 128 x 64 pixlar. Displayen var uppdelad i två likadana displayer med 64 x 64 pixlar. Varje halva hade 64 adresser i y-led och 8 register i x-led. Givet en adress i y-led kunde man för varje adress x-led kontrollera 8 pixlar. På detta sätt kunde man kontrollera alla skärmens pixlar. Pixlarna hade två lägen. Antingen tända (svart pixel) eller släckta (vit pixel).

# Varistor

En varistor är en variabel resistor. Genom att reglera spänningen till en av displayens pinnar kan man styra dess kontrast och svärta.

# Knapp

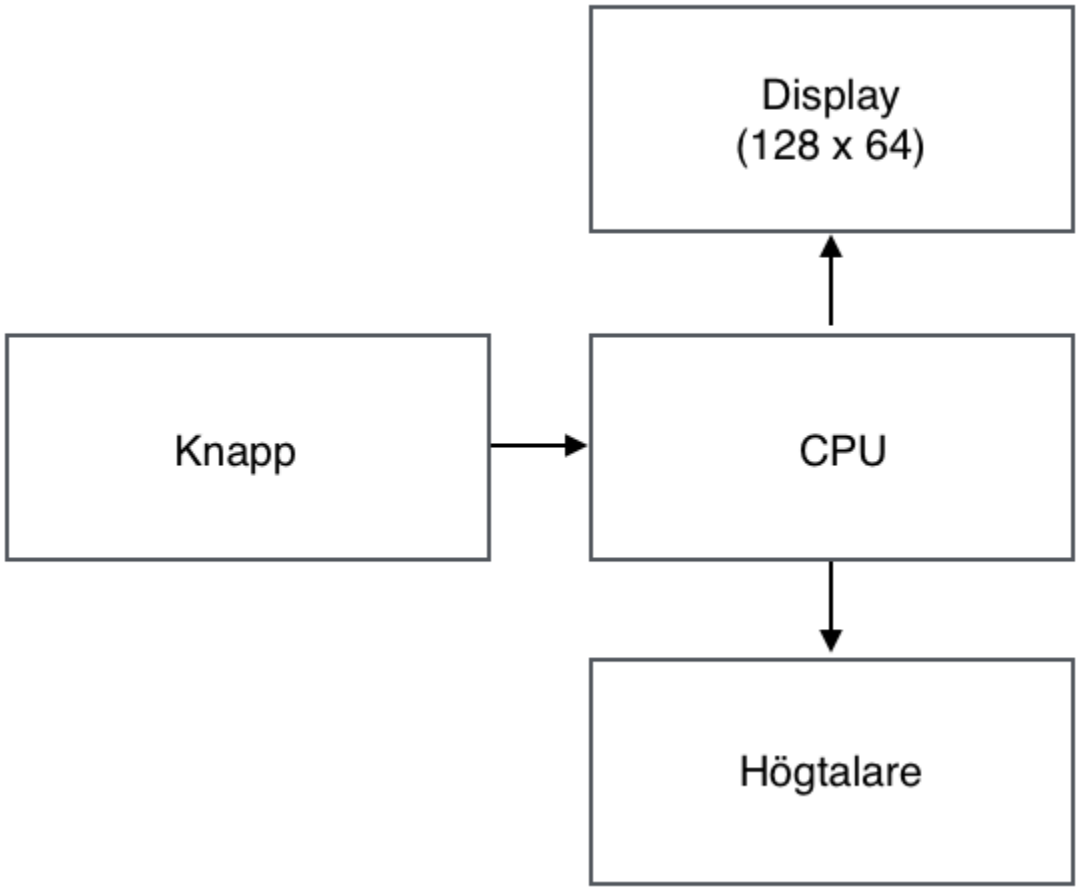
Spelets enda insignal är en röd knapp. Den här knappen används för att starta spelet, styra fågeln och starta om spelet när fågeln dör.

# Högtalare

För att generera ljud är en högtalare inkopplad till en av pinnarna på processorns B-port. Fördelen med dessa porart är att de kan generera önskade utsignaler med hjälp av pulsbreddsmodellering (PWM). Genom att slå på och av spänningen snabbare än vad högtalaren kan urskilja, kan processorn skapa en kontinuerligt varierbar effekt som omvandlas till ljud i högtalaren.  (Wikipedia, 2014b)

Eftersom alla toner är någon form av sinusvåg, har en sinusvåg samplats i filen sin.h. Denna används av processor för att spela upp de önskade tonerna.

## Blockschema




# Mjukvara

Programmet till mikrokontrollen är skrivet i C. Programmet skrevs i editorn Atmel Studio, version 6. Med hjälp av en J-TAG kopplad till processorns C-portar kunde sedan programmet laddas ned till microkontrollern. Huvudprogrammet består av en while-loop som exekverar metoder beroende på vilken fas spelet befinner sig i. Dessa tre faser är startfasen, spelfasen och slutfasen.

I startfasen initeras displayen och alla pixar sätts till vita. Alla spelberoende variabler nollställs och startskärmen ritas upp. Ett ljud spelas upp för att signalera att programmet är redo för start. När användaren trycker på knappen, inleds nästa fas.

Spelfasen är beroende av tick, alltså en loop inuti ett program där flera sekvenser inträffar samtidigt. Under en tick, kontrolleras fågelns tillstånd. Har den krockat med ett rör eller

förflyttat sig utanför displayen, avslutas spelfasen och slutfasen inleds. Är fågeln vid liv lyssnar programmet efter användarinmatning på knappen. Är knappen intryckt förflyttas fågeln uppåt, annars åker fågeln nedåt. Under varje tick förflyttas även rören ett steg närmare fågeln.

Om fågeln dör inleds slutfasen. Ett dödsljud spelas upp och displayen töms för att göra plats åt användarens samlade poängsumma. Ifall användaren klickar på kanppen igen, startar spelet om i startfasen.

En metod för att manipulera en enskild pixel skapades. För att den inte skulle påverka någon av de andra åtta pixlarna som låg på samma band så användes en matris föra att hämta information från de andra sju pixlarna. Alltså skrevs pixlarna om på nytt men informationen bibehölls genom att läsa av data ifrån matrisen.

Alla figurer som användes i spelet har en egen metod som anropar setpixel metoden flertalet gånger så att pixlarna bildade figuren.

Skrämen har ingen egen funktion för att hantera bokstäver/nummer. Ett typsnitt med de bokstäver och nummer skapades därför för att kunna skriva ut dessa på skrämen på samma sätt som alla andra figurer.

Hela källkoden kan begrundas bland bilagorna.

# Arbetsprocessen

## Planering

Först utfördes en kravspecifikation på vad spelkonsolen/spelet skulle utföra. Utefter denna valdes de olika komponenterna ut. Ett kopplingschema upprättades med komponenterna samt deras kopplingar.

## Montering av hårdvara

Monteringen utfördes på ett mönsterkort där komponenterna virades med en virpistol alternativt löddes fast på kortet. Efter monteringen testades komponenterna/kopplingarna genom att skriva få rader test kod och genom att manuellt styra portarna i Atmel Studio i Debug mode.

De ursprungliga kopplingarna var korrekta, de enda modifikationerna som gjorts efter det första skedet var bortagning av komponenter som visade sig vara onödiga och kondensatorer för att motverka störningar. Störningarna var inte något problem men var en del av felsökning utav skärmen.

## Programmering

För att programmera mikrokontrollen användes programmet Atmel Studio, version 6. Programmet laddades sedan ner till processor via en JTAG kopplad till processorns C-portar.

Fördelen med en JTAG är att den kan styra register och specifika pinnar på mikrokontrollern direkt, vilket underlättade programmeringen avsevärt. Tack vare debug-funktionen och möjligheten att stega igenom programmet hos JTAGen, kunde programmet felsökas då programmet inte uppförde sig som önskat.

## Problem

Det som har medfört överlägset mest problem är programmeringen av displayen. För att kunna skriva spelet behövdes som grund två funktioner kopplande till displayen. En för att tända en pixel och en för att släcka en pixel. Först tog det lång tid att förstå hur man skulle gå tillväga för att adressera sig fram till en specifik pixel given en x-koordinat och en y-koordinat.

När adresseringen var löst fungerade metoderna fortfarande inte som de skulle. Skärmen tände pixlar, släckte pixlar och stängde av sig utan att den fått instruktioner att göra det. Först antogs skärmen vara trasig men detta visade sig inte vara fallet. Sedan antogs störningar vara anledningen varpå kondensatorer kopplades till både skärmen och till mikrokontrollern men även detta hjälpte inte. Till slut visade det sig att felet var en inställningen i Atmel Studio. Inställningen optimerade koden vilket medfor att portarn växlade mellan ettor och nollor i en annan ordning än den tänkta. Inställningen togs bort och problemet var löst.

I början av projektet antogs det att ljudet skulle vara det svåraste momentet. Det här visade sig vara felaktigt då PWM kunde användas vilket gjorde att ett bra tv-spelsljud kunde uppnås utan att använda vare sig den externa D/A omvandlaren eller mikrofonen.

# Resultat

Resultatet är en spelkonsoll som fungerar enligt vår kravspecifikation. Ljud grafik och styrning med hjälp av knappen fungerar som tänkt. Hårdvaran är inte optimal för denna typ av actionspel. Eftersom både fågeln och rören rör på sig hela tiden så måste skärmen ta bort dem och rita upp den väldigt ofta. Detta ger intrycket av objekten på skärmen blinkar/flimrar svagt. Sedan är det inte optimalt att spela upp ljud under spelets gång då spelet står stilla och väntar medan ljudet spelas upp. Dessa är dock imperfektioner tillräckligt små i vårt fall för att inte väsentligt försämra spelupplevelsen.

# Slutsats

Det här projektet har varit mycket givanade. Projektet har medfört en förståelse i hur hårdvara fungerar samt förståelse i kopplingen mellan hårdvara och mjukvara.

Vi är mycket nöjda med projektet. Den känslan som vi tar med oss från kursen är att ingenting är omöjligt och att datablad inte är så krångliga, bara man sätter sig ner och läser dem. Vi har fått mer förståelse för hur mikrokontrollers fungerar och hur saker och ting fungerar på lågnivå, att allt vi gör egentligen bara består av ettor och nollor, ström på och ström av.

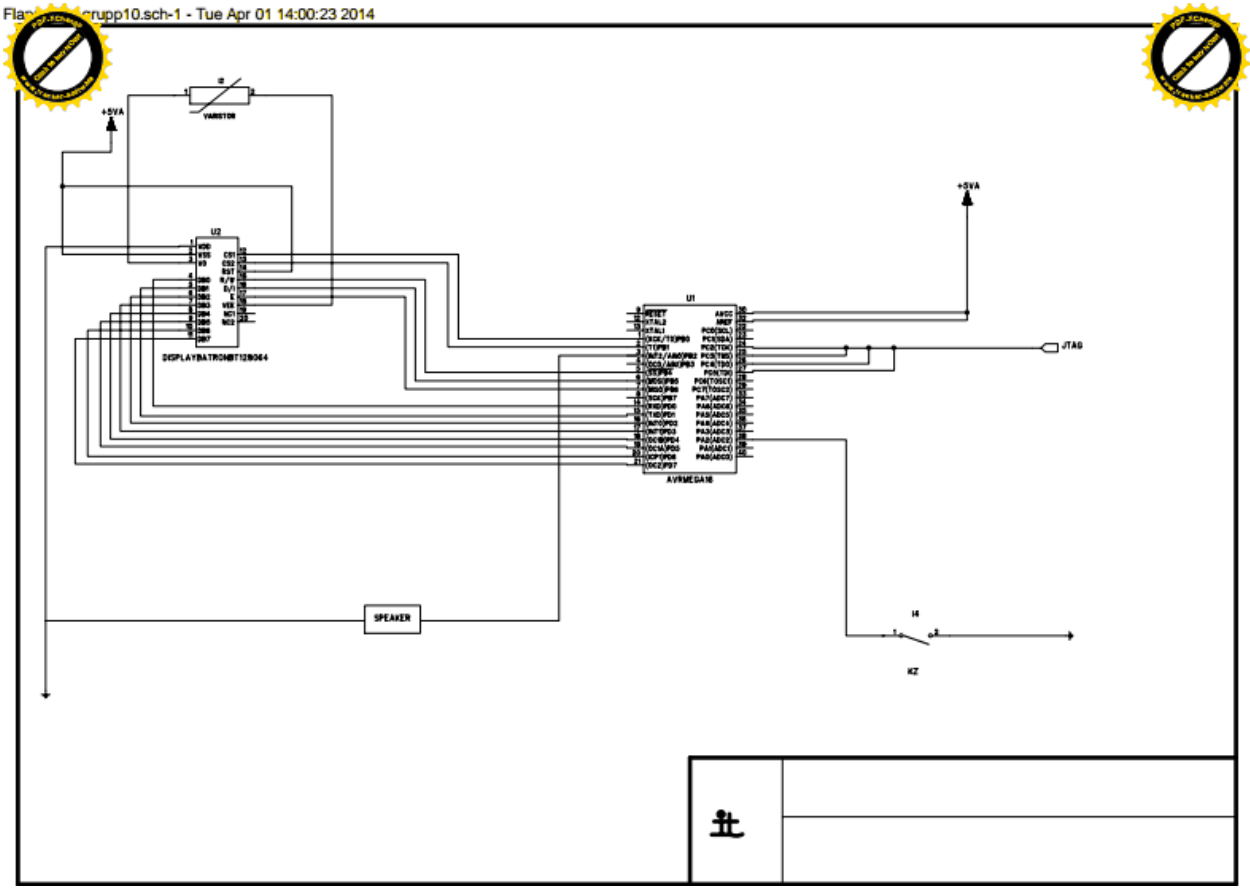
# Referenser

Wikipeda (2014a). *Pulsbreddsmodulering*. http://sv.wikipedia.org/wiki/Pulsbreddsmodulering [2014-04-01]

Wikipeda (2014b). *Flappy Bird*. http://sv.wikipedia.org/wiki/Flappy_Bird [2014-04-01]

# Bilagor

# Kopplingsschema



# Källkod

```
/*
 * FlappyBird.c
 *
 * Created: 2014-03-21 15:11:35
 *  Author: digpi10
 */

#include <avr/io.h>
#include <util/delay.h>

#include "display.h"
#include "draw.h"
#include "drawings.h"
#include "sound.h"

short int mainButtonPressed;
short int menuButtonPressed;

short int birdIsDead;
short int collision;
short int score = 0;

short int tickDelay = 50;
short int gamePhase;

struct Bird {
    int x;
        int y;
};

struct Tube {
        int x;
        int y;
};

struct Tube tubeArray[3];
struct Bird bird;

struct Tube nullTube;

short int tickCount = 0;
short int createNewTubePace = 40;
short int numberOfTubes = 0;

int main(void)
{
        startGame();
        while(1){
        }
}

void startGame() {
        setup();
        while (1) {
                switch(gamePhase) {
                        case 0:
                                startPhase();
                                break;
                        case 1:
                                game();
                                break;
                        case 2:
                                showResults();
                                break;
                        default:
                        break;
                }
```

```
        }
}

void game() {
         birdIsDead = 0;
         while (birdIsDead != 1) {
                 tick();
                 _delay_ms(tickDelay);
         }
         gamePhase = 2;
}

void tick() {
        checkCollision();
        if (collision == 1) {
                birdIsDead = 1;
        } else {
                readMainButton();
                if (mainButtonPressed) {
                        moveBirdUp();
                } else {
                        moveBirdDown();
                }
                moveTubes();
                scoreCount();
                if (tickCount > createNewTubePace) {
                        if (numberOfTubes < 2) {
                                numberOfTubes++;
                                createNewTube(numberOfTubes);
                        }
                        tickCount = 0;
                }
                tickCount++;

        }
}

void scoreCount(){
        for (int k = 0; k < 3; k++) {
                if ( tubeArray[k].y == (bird.y + 15)) {
                        score++;
                        jumpSound();
                }
        }
}

void showResults() {
        removeBird();
        setScore(paintBlack,score);
        drawFrame(paintWhite);
        for (int k = 0; k < 3; k++) {
                removeTube(tubeArray[k].x,tubeArray[k].y);
        }
        deathSound();
        while(1) {
                readMainButton();
                if(mainButtonPressed) {
                        startGame();
                }
        }

}

void checkCollision() {
        if(bird.x > 51) {
                collision = 1;
        } else if (bird.x < 1) {
                collision = 1;
        } else {
                for (int k = 0; k < numberOfTubes; k++) {
                        if((tubeArray[k].y+6) > bird.y) {
```

```
                                if (tubeArray[k].y < (bird.y + 15)) {
                                        if(tubeArray[k].x + 2 > bird.x) {
                                                collision = 1;
                                        }
                                        if ((tubeArray[k].x + 2 + 30) < bird.x + 12) {
                                                collision = 1;
                                        }

                                }
                        }
                }
        }
}

void moveBirdUp() {
        removeBird();
        bird.x += 1;
        drawBird();
}

void moveBirdDown() {
        removeBird();
        bird.x -= 1;
        drawBird();
}

void moveTubes() {
        removeTube(tubeArray[0].x,tubeArray[0].y);
        tubeArray[0].y++;
        if (tubeArray[0].y > 121) {
                tubeArray[0].y = 0;
        } else {
                drawTube(tubeArray[0].x,tubeArray[0].y);
        }
        if (numberOfTubes > 0) {
                removeTube(tubeArray[1].x,tubeArray[1].y);
                tubeArray[1].y++;
                if (tubeArray[1].y > 121) {
                        tubeArray[1].y = 0;
                } else {
                        drawTube(tubeArray[1].x,tubeArray[1].y);
                }
        }
        if (numberOfTubes > 1) {
                removeTube(tubeArray[2].x,tubeArray[2].y);
                tubeArray[2].y++;
                if (tubeArray[2].y > 121) {
                        tubeArray[2].y = 0;
                } else {
                        drawTube(tubeArray[2].x,tubeArray[2].y);
                }
        }
}

void startPhase() {
        bird.x = 20;
        bird.y = 84;
        drawStartScreen();
        createNewTube(0);
        startSound();
        while(1){
                readMainButton();
                if (mainButtonPressed) {
                         gamePhase = 1;
                         setTapToStart(paintWhite);
                         drawFrame(paintWhite);
                         return;
                }
        }
}
```

```
void createNewTube(int n) {

		struct Tube tube;
		tubeArray[n] = tube;
		tubeArray[n].x = 10 + (5 * n);
		tubeArray[n].y = 1;
		drawTube(tubeArray[n].x, tubeArray[n].y);

}

void setup() {
	DDRD = 0xFF;
	DDRB = 0xFF;
	dispOn();
	emptyDisplay();
	gamePhase = 0;
	score = 0;
	collision = 0;
	numberOfTubes = 0;
	tickCount = 0;
}

void drawBird() {
	setBird(bird.x, bird.y, paintBlack);
}

void removeBird() {
	setBird(bird.x, bird.y, paintWhite);
}

void readMainButton() {
	short int status = PINA | 0xF3;
	if ( status == 0xF3) {
		status = PINA | 0xFB;
		if (status == 0xFB) {
			mainButtonPressed = 1; // knappen är nedtryck
		}
	} else {
		mainButtonPressed = 0;// knappen är inte nedtryckt
	}
}
void readMenuButton() {
	short int status = PINA | 0xFD;
	if ( status == 0xFD) {
		menuButtonPressed = 1; // knappen är nedtryck
	} else {
		menuButtonPressed = 0; // knappen är inte nedtryckt
	}
}

/*
 * display.h
 *
 * Created: 2014-03-21 15:11:35
 *  Author: digpi10
 */


char display[8][128];

void dispOn() {
	DDRD = 0xFF;
	PORTD = 0x3F;
	PORTB = 0xCD;
	PORTB = PORTB & 0xBF;

	PORTB = 0xCE;
	PORTB = PORTB & 0xBF;
}

void dispOff() {
```

```
	DDRD = 0xFF;
	PORTD = 0x3E;
	PORTB = 0xCD;
	PORTB = PORTB & 0xBF;

	PORTB = 0xCE;
	PORTB = PORTB & 0xBF;
}
/*
 * draw.h
 *
 * Created: 2014-03-21 15:11:35
 *  Author: digpi10
 */

short int paintBlack = 1;
short int paintWhite = 0;

void setPixel(short int x, short int y, short int operation) {
	PORTB = PORTB | 0x03;
	PORTD = 0x00;
	PORTB = PORTB | 0x10;
	int yPos;
	int xPos = x;
	if( y < 64) {
		PORTB = PORTB & 0xFD;
		yPos = y;
	} else {
		PORTB = PORTB & 0xFE;
		yPos = y - 64;
	}
	PORTB = PORTB&0xCF;
	setXAddress(x);
	setYAddress(yPos);
	paintPixel(xPos, y, operation);
}

void toggleESignal() {
	PORTB = PORTB | 0x40;
	PORTB = PORTB & 0xBF;
	PORTB = PORTB | 0x40;
}

void setXAddress(short int x) {
	short int xPos = x / 8;
	PORTD = 0xB8 | xPos;
	toggleESignal();
}

void setYAddress(short int y) {
	PORTD = 0x40 | y;
	toggleESignal();
}

void paintPixel(short int x, short int y, short int operation) {
	PORTB = PORTB | 0xEC;
	short int xAddress = (x / 8);
	short int pinOut = (x & 0x07); // genererar 8 bitar där den n:e biten (n = x % 8) är en
etta.
	if (operation == paintBlack) {
		display[xAddress][y] = display[xAddress][y] | (0x01 << pinOut);
		PORTD = display[xAddress][y];
	} else {
		display[xAddress][y] = display[xAddress][y] & (0xff - ((0x01 << pinOut)));
		PORTD = display[xAddress][y];
	}
	toggleESignal();
}

void drawLine(int start, int stop) {
```

```
}

void emptyDisplay() {
	for(int x = 0; x < 8; x++) {
		for(int y = 0; y < 128; y++) {
			setPixel(x*8 + 0,y,paintWhite);
			setPixel(x*8 + 1,y,paintWhite);
			setPixel(x*8 + 2,y,paintWhite);
			setPixel(x*8 + 3,y,paintWhite);
			setPixel(x*8 + 4,y,paintWhite);
			setPixel(x*8 + 5,y,paintWhite);
			setPixel(x*8 + 6,y,paintWhite);
			setPixel(x*8 + 7,y,paintWhite);
			display[x/8][y] = 0x00;
		}
	}
}

void setPixel2(int x, int y, short int operation) {
	char xAddress = (x / 8);
	short int pinOut = (x & 0x07);
	if(operation == paintBlack) {
		display[xAddress][y] = display[xAddress][y] | (0x01 << pinOut);
	} else {
		display[xAddress][y] = display[xAddress][y] & (0xff - ((0x01 << pinOut)));
	}
}

void reDraw() {
	PORTB = PORTB | 0xEC;
	PORTB = PORTB & 0xEF;
	for(int x = 0; x < 64; x++) {
		for(int y = 0; y < 128; y++) {
			PORTB = PORTB | 0x03;
			char xAddress = (x / 8);
			char yPos;
			if( y < 64) {
				PORTB = PORTB & 0xFD;
				yPos = y;
			} else {
				PORTB = PORTB & 0xFE;
				yPos = y - 64;
			}
			setXAddress(x);
			setYAddress(yPos);
			PORTD = display[xAddress][y];

			toggleESignal();
			}
		}
}
/*
 * drawings.h
 *
 * Created: 2014-03-21 15:11:35
 *  Author: digpi10
 */

short int tubeGap = 30;

void setBird(char x, char y, short int operation){
	setPixel(x+0 ,y+6,operation);
	setPixel(x+0 ,y+7,operation);
	setPixel(x+0 ,y+8,operation);
	setPixel(x+0 ,y+9,operation);
	setPixel(x+0 ,y+10,operation);

	setPixel(x+1, y+2,operation);
	setPixel(x+1, y+3,operation);
	setPixel(x+1, y+4,operation);
	setPixel(x+1, y+5,operation);
```

```
        setPixel(x+1, y+11,operation);
        setPixel(x+1, y+12,operation);

        setPixel(x+2, y+1,operation);
        setPixel(x+2, y+13,operation);

        setPixel(x+3, y+1,operation);
        setPixel(x+3, y+2,operation);
        setPixel(x+3, y+3,operation);
        setPixel(x+3, y+4,operation);
        setPixel(x+3, y+5,operation);
        setPixel(x+3, y+11,operation);
        setPixel(x+3, y+12,operation);
        setPixel(x+3, y+13,operation);

        setPixel(x+4, y+0,operation);
        setPixel(x+4, y+10,operation);
        setPixel(x+4, y+14,operation);

        setPixel(x+5, y+1,operation);
        setPixel(x+5, y+2,operation);
        setPixel(x+5, y+3,operation);
        setPixel(x+5, y+4,operation);
        setPixel(x+5, y+5,operation);
        setPixel(x+5, y+9,operation);
        setPixel(x+5, y+15,operation);

        setPixel(x+6, y+2,operation);
        setPixel(x+6, y+6,operation);
        setPixel(x+6, y+9,operation);
        setPixel(x+6, y+15,operation);

        setPixel(x+7, y+2,operation);
        setPixel(x+7, y+4,operation);
        setPixel(x+7, y+7,operation);
        setPixel(x+7, y+10,operation);
        setPixel(x+7, y+15,operation);

        setPixel(x+8, y+2,operation);
        setPixel(x+8, y+4,operation);
        setPixel(x+8, y+7,operation);
        setPixel(x+8, y+11,operation);
        setPixel(x+8, y+12,operation);
        setPixel(x+8, y+13,operation);
        setPixel(x+8, y+14,operation);

        setPixel(x+9, y+3,operation);
        setPixel(x+9, y+7,operation);
        setPixel(x+9, y+12,operation);

        setPixel(x+10, y+4,operation);
        setPixel(x+10, y+6,operation);
        setPixel(x+10, y+10,operation);
        setPixel(x+10, y+11,operation);

        setPixel(x+11, y+5,operation);
        setPixel(x+11, y+6,operation);
        setPixel(x+11, y+7,operation);
        setPixel(x+11, y+8,operation);
        setPixel(x+11, y+9,operation);
}

void drawFrame(short int operation) {
        for(char x = 0; x < 64 ; x++) {
                setPixel(x, 0, operation);
                setPixel(x, 127, operation);
        }
        for(char y = 0; y < 128; y++) {
                setPixel(0, y, operation);
                setPixel(63,y, operation);
        }
```

```
}
void drawA(short int x, short int y, short int operation) {
        setPixel(x+0,y+0,operation);
        setPixel(x+0,y+4,operation);
        setPixel(x+1,y+0,operation);
        setPixel(x+1,y+4,operation);
        setPixel(x+2,y+0,operation);
        setPixel(x+2,y+1,operation);
        setPixel(x+2,y+2,operation);
        setPixel(x+2,y+3,operation);
        setPixel(x+2,y+4,operation);
        setPixel(x+3,y+0,operation);
        setPixel(x+3,y+4,operation);
        setPixel(x+4,y+1,operation);
        setPixel(x+4,y+2,operation);
        setPixel(x+4,y+3,operation);
}
void drawT(short int x, short int y, short int operation) {
        setPixel(x+0,y+2,operation);
        setPixel(x+1,y+2,operation);
        setPixel(x+2,y+2,operation);
        setPixel(x+3,y+2,operation);
        setPixel(x+4,y+0,operation);
        setPixel(x+4,y+1,operation);
        setPixel(x+4,y+2,operation);
        setPixel(x+4,y+3,operation);
        setPixel(x+4,y+4,operation);
}
void drawP(short int x, short int y, short int operation) {
        setPixel(x+0,y+4,operation);
        setPixel(x+1,y+4,operation);
        setPixel(x+2,y+1,operation);
        setPixel(x+2,y+2,operation);
        setPixel(x+2,y+3,operation);
        setPixel(x+2,y+4,operation);
        setPixel(x+3,y+4,operation);
        setPixel(x+3,y+0,operation);
        setPixel(x+4,y+1,operation);
        setPixel(x+4,y+2,operation);
        setPixel(x+4,y+3,operation);
        setPixel(x+4,y+4,operation);
}
void drawO(short int x, short int y, short int operation) {
        setPixel(x+0,y+1,operation);
        setPixel(x+0,y+2,operation);
        setPixel(x+0,y+3,operation);
        setPixel(x+1,y+0,operation);
        setPixel(x+1,y+4,operation);
        setPixel(x+2,y+0,operation);
        setPixel(x+2,y+4,operation);
        setPixel(x+3,y+0,operation);
        setPixel(x+3,y+4,operation);
        setPixel(x+4,y+1,operation);
        setPixel(x+4,y+2,operation);
        setPixel(x+4,y+3,operation);
}
void drawS(short int x, short int y, short int operation) {
        setPixel(x+0,y+1,operation);
        setPixel(x+0,y+2,operation);
        setPixel(x+0,y+3,operation);
        setPixel(x+0,y+4,operation);
        setPixel(x+1,y+0,operation);
        setPixel(x+2,y+1,operation);
        setPixel(x+2,y+2,operation);
        setPixel(x+2,y+3,operation);
        setPixel(x+3,y+4,operation);
        setPixel(x+4,y+0,operation);
        setPixel(x+4,y+1,operation);
        setPixel(x+4,y+2,operation);
        setPixel(x+4,y+3,operation);
```

```
}
void drawR(short int x, short int y, short int operation) {
	setPixel(x+0,y+0,operation);
	setPixel(x+0,y+4,operation);
	setPixel(x+1,y+0,operation);
	setPixel(x+1,y+4,operation);
	setPixel(x+2,y+1,operation);
	setPixel(x+2,y+2,operation);
	setPixel(x+2,y+3,operation);
	setPixel(x+2,y+4,operation);
	setPixel(x+3,y+0,operation);
	setPixel(x+3,y+4,operation);
	setPixel(x+4,y+1,operation);
	setPixel(x+4,y+2,operation);
	setPixel(x+4,y+3,operation);
	setPixel(x+4,y+4,operation);
}
void drawE(short int x, short int y, short int operation){
	setPixel(x+0,y+0,operation);
	setPixel(x+0,y+1,operation);
	setPixel(x+0,y+2,operation);
	setPixel(x+0,y+3,operation);
	setPixel(x+0,y+4,operation);
	setPixel(x+1,y+4,operation);
	setPixel(x+2,y+1,operation);
	setPixel(x+2,y+2,operation);
	setPixel(x+2,y+3,operation);
	setPixel(x+2,y+4,operation);
	setPixel(x+3,y+4,operation);
	setPixel(x+4,y+0,operation);
	setPixel(x+4,y+1,operation);
	setPixel(x+4,y+2,operation);
	setPixel(x+4,y+3,operation);
	setPixel(x+4,y+4,operation);
}
void drawC(short int x, short int y, short int operation){
	setPixel(x+0,y+1,operation);
	setPixel(x+0,y+2,operation);
	setPixel(x+0,y+3,operation);
	setPixel(x+1,y+0,operation);
	setPixel(x+1,y+4,operation);
	setPixel(x+2,y+4,operation);
	setPixel(x+3,y+0,operation);
	setPixel(x+3,y+4,operation);
	setPixel(x+4,y+1,operation);
	setPixel(x+4,y+2,operation);
	setPixel(x+4,y+3,operation);
}
void setTapToStart(int operation) {
	drawT(5,97,operation);
	drawA(5,91,operation);
	drawP(5,85,operation);
	drawT(5,74,operation);
	drawO(5,68,operation);
	drawS(5,57,operation);
	drawT(5,51,operation);
	drawA(5,45,operation);
	drawR(5,39,operation);
	drawT(5,33,operation);
}

void drawStartScreen() {
	drawFrame(paintBlack);
	drawBird();
	setTapToStart(paintBlack);
}

void setScore(short int operation, short int score){
	drawS(25,80,operation);
	drawC(25,74,operation);
	drawO(25,68,operation);
```

```
        drawR(25,62,operation);
        drawE(25,56,operation);
        setPixel(28,54,operation);
        setPixel(26,54,operation);

        short int firstDigit = score % 10;
        short int secondDigit = score / 10;

        drawNumber(25,43,operation,firstDigit);
        drawNumber(25,47,operation,secondDigit);
}
void drawNumber(short int x, short int y, short int operation, short int digit) {
                        switch(digit) {
                        case 0:
                                draw0(x, y, operation);
                                break;
                        case 1:
                                draw1(x, y, operation);
                                break;
                        case 2:
                                draw2(x, y, operation);
                                break;
                        case 3:
                                draw3(x, y, operation);
                                break;
                        case 4:
                                draw4(x, y, operation);
                                break;
                        case 5:
                                draw5(x, y, operation);
                                break;
                        case 6:
                                draw6(x, y, operation);
                                break;
                        case 7:
                                draw7(x, y, operation);
                                break;
                        case 8:
                                draw8(x, y, operation);
                                break;
                        case 9:
                                draw9(x, y, operation);
                                break;
                        default:
                        break;
                }
}
void draw0(short int x, short int y, short int operation){
        setPixel(x+0,y+0,paintBlack);
        setPixel(x+1,y+0,paintBlack);
        setPixel(x+2,y+0,paintBlack);
        setPixel(x+3,y+0,paintBlack);
        setPixel(x+4,y+0,paintBlack);
        setPixel(x+0,y+1,paintBlack);
        setPixel(x+4,y+1,paintBlack);
        setPixel(x+0,y+2,paintBlack);
        setPixel(x+1,y+2,paintBlack);
        setPixel(x+2,y+2,paintBlack);
        setPixel(x+3,y+2,paintBlack);
        setPixel(x+4,y+2,paintBlack);
}
void draw1(short int x, short int y, short int operation){
        setPixel(x+0,y+0,paintBlack);
        setPixel(x+1,y+0,paintBlack);
        setPixel(x+2,y+0,paintBlack);
        setPixel(x+3,y+0,paintBlack);
        setPixel(x+4,y+0,paintBlack);
}
void draw2(short int x, short int y, short int operation){
        setPixel(x+0,y+0,paintBlack);
        setPixel(x+0,y+1,paintBlack);
```

```
        setPixel(x+0,y+2,paintBlack);
        setPixel(x+1,y+2,paintBlack);
        setPixel(x+2,y+0,paintBlack);
        setPixel(x+2,y+1,paintBlack);
        setPixel(x+2,y+2,paintBlack);
        setPixel(x+3,y+0,paintBlack);
        setPixel(x+4,y+0,paintBlack);
        setPixel(x+4,y+1,paintBlack);
        setPixel(x+4,y+2,paintBlack);
}
void draw3(short int x, short int y, short int operation){
        setPixel(x+0,y+0,paintBlack);
        setPixel(x+0,y+1,paintBlack);
        setPixel(x+0,y+2,paintBlack);
        setPixel(x+1,y+0,paintBlack);
        setPixel(x+2,y+0,paintBlack);
        setPixel(x+2,y+1,paintBlack);
        setPixel(x+2,y+2,paintBlack);
        setPixel(x+3,y+0,paintBlack);
        setPixel(x+4,y+0,paintBlack);
        setPixel(x+4,y+1,paintBlack);
        setPixel(x+4,y+2,paintBlack);
}
void draw4(short int x, short int y, short int operation){
        setPixel(x+0,y+0,paintBlack);
        setPixel(x+1,y+0,paintBlack);
        setPixel(x+2,y+0,paintBlack);
        setPixel(x+2,y+1,paintBlack);
        setPixel(x+2,y+2,paintBlack);
        setPixel(x+3,y+0,paintBlack);
        setPixel(x+3,y+2,paintBlack);
        setPixel(x+4,y+0,paintBlack);
        setPixel(x+4,y+2,paintBlack);
}
void draw5(short int x, short int y, short int operation){
        setPixel(x+0,y+0,paintBlack);
        setPixel(x+0,y+1,paintBlack);
        setPixel(x+0,y+2,paintBlack);
        setPixel(x+1,y+0,paintBlack);
        setPixel(x+2,y+0,paintBlack);
        setPixel(x+2,y+1,paintBlack);
        setPixel(x+2,y+2,paintBlack);
        setPixel(x+3,y+2,paintBlack);
        setPixel(x+4,y+0,paintBlack);
        setPixel(x+4,y+1,paintBlack);
        setPixel(x+4,y+2,paintBlack);
}
void draw6(short int x, short int y, short int operation){
        setPixel(x+0,y+0,paintBlack);
        setPixel(x+0,y+1,paintBlack);
        setPixel(x+0,y+2,paintBlack);
        setPixel(x+1,y+0,paintBlack);
        setPixel(x+1,y+2,paintBlack);
        setPixel(x+2,y+0,paintBlack);
        setPixel(x+2,y+1,paintBlack);
        setPixel(x+2,y+2,paintBlack);
        setPixel(x+3,y+2,paintBlack);
        setPixel(x+4,y+0,paintBlack);
        setPixel(x+4,y+1,paintBlack);
        setPixel(x+4,y+2,paintBlack);
}
void draw7(short int x, short int y, short int operation){
        setPixel(x+0,y+0,paintBlack);
        setPixel(x+1,y+0,paintBlack);
        setPixel(x+2,y+0,paintBlack);
        setPixel(x+3,y+0,paintBlack);
        setPixel(x+4,y+0,paintBlack);
        setPixel(x+4,y+1,paintBlack);
        setPixel(x+4,y+2,paintBlack);
}
void draw8(short int x, short int y, short int operation){
```

```
        setPixel(x+0,y+0,paintBlack);
        setPixel(x+0,y+1,paintBlack);
        setPixel(x+0,y+2,paintBlack);
        setPixel(x+1,y+0,paintBlack);
        setPixel(x+1,y+2,paintBlack);
        setPixel(x+2,y+0,paintBlack);
        setPixel(x+2,y+1,paintBlack);
        setPixel(x+2,y+2,paintBlack);
        setPixel(x+3,y+0,paintBlack);
        setPixel(x+3,y+2,paintBlack);
        setPixel(x+4,y+0,paintBlack);
        setPixel(x+4,y+1,paintBlack);
        setPixel(x+4,y+2,paintBlack);
}
void draw9(short int x, short int y, short int operation){
        setPixel(x+0,y+0,paintBlack);
        setPixel(x+0,y+1,paintBlack);
        setPixel(x+0,y+2,paintBlack);
        setPixel(x+1,y+0,paintBlack);
        setPixel(x+2,y+0,paintBlack);
        setPixel(x+2,y+1,paintBlack);
        setPixel(x+2,y+2,paintBlack);
        setPixel(x+3,y+0,paintBlack);
        setPixel(x+3,y+2,paintBlack);
        setPixel(x+4,y+0,paintBlack);
        setPixel(x+4,y+1,paintBlack);
        setPixel(x+4,y+2,paintBlack);
}



void drawTubeUp(short int y, short int length, short int operation) {
        short int x = length;

        for(int k = 0; k < x; k++) {
                setPixel(k, y+1, operation);
                setPixel(k, y+4, operation);
        }

        setPixel(x, y+0, operation);
        setPixel(x, y+1, operation);
        setPixel(x, y+4, operation);
        setPixel(x, y+5, operation);
        setPixel(x+1, y+0, operation);
        setPixel(x+1, y+5, operation);
        setPixel(x+2, y+0, operation);
        setPixel(x+2, y+1, operation);
        setPixel(x+2, y+2, operation);
        setPixel(x+2, y+3, operation);
        setPixel(x+2, y+4, operation);
        setPixel(x+2, y+5, operation);

}
void drawTubeDown(short int y, short int length, short int operation) {
        short int x = (63 - length);
        for(int k = 63; k > x; k--) {
                setPixel(k, y+1, operation);
                setPixel(k, y+4, operation);
        }
        setPixel(x, y+0, operation);
        setPixel(x, y+1, operation);
        setPixel(x, y+4, operation);
        setPixel(x, y+5, operation);
        setPixel(x-1, y+0, operation);
        setPixel(x-1, y+5, operation);
        setPixel(x-2, y+0, operation);
        setPixel(x-2, y+1, operation);
        setPixel(x-2, y+2, operation);
        setPixel(x-2, y+3, operation);
        setPixel(x-2, y+4, operation);
        setPixel(x-2, y+5, operation);
```

```
}
void drawTube(char x, char y){
       drawTubeUp(y, x-3, paintBlack);
       drawTubeDown(y, 64-x-tubeGap-3, paintBlack);
}

void removeTube(char x, char y){
       drawTubeUp(y, x-3, paintWhite);
       drawTubeDown(y, 64-x-tubeGap-3, paintWhite);
}

#include <avr/pgmspace.h>

#include "sin.h"

/*

Function To Initialize TIMER0 In Fast
PWWM Mode.

*/
void InitPWM()
{
   TCCR0|=(1<<WGM00)|(1<<WGM01)|(1<<COM01)|(1<<CS00);
   DDRB|=(1<<PB3);
}

void deathSound()
{
   uint8_t i;

   InitPWM();

      uint8_t delay,n;

      for(delay=1;delay<=50;delay++)
      {
         for(n=0;n<(51-delay);n++)
         {
            for(i=0;i<=10;i++)
            {
               OCR0=pgm_read_byte_near(sine+i);
               _delay_loop_2(delay);

            }
         }
      }
      for(delay=50;delay>=2;delay--)
      {
         for(n=0;n<(51-delay);n++)
         {
            for(i=0;i<=5;i++)
            {
               OCR0=pgm_read_byte_near(sine+i);
               _delay_loop_2(delay);
            }
         }
      }

   }
void startSound()
{
   uint8_t i;

   InitPWM();

      uint8_t delay,n;

      for(delay=1;delay<=50;delay++)
      {
```

```
         for(n=0;n<(51-delay);n++)
         {
            for(i=0;i<=20;i++)
            {
               OCR0=pgm_read_byte_near(sine+i);
               _delay_loop_2(delay);

            }
         }
      }
      for(delay=50;delay>=2;delay--)
      {
         for(n=0;n<(51-delay);n++)
         {
            for(i=0;i<=20;i++)
            {
               OCR0=pgm_read_byte_near(sine+i);
               _delay_loop_2(delay);
            }
         }
      }

   }

void jumpSound()
{
   uint8_t I;
   InitPWM();
      uint8_t delay,n;
      for(delay=1;delay<=20;delay++){
         for(n=0;n<(51-delay);n++){
            for(i=0;i<=3;i++){
               OCR0=pgm_read_byte_near(sine+i);
               _delay_loop_2(delay);
            }
         }
      }
      for(delay=20;delay>=2;delay--){
         for(n=0;n<(51-delay);n++){
            for(i=0;i<=4;i++){
               OCR0=pgm_read_byte_near(sine+i);
               _delay_loop_2(delay);
            }
         }
      }
   }

#include <avr/pgmspace.h>

uint8_t sine[256]={127, 130, 133, 136, 139, 143, 146, 149, 152, 155, 158, 161, 164, 167, 170,
173, 176, 178, 181, 184, 187, 189, 192, 195, 197, 200, 203, 205, 207, 210, 212, 214, 217, 219,
221, 223, 225, 227, 229, 231, 232, 234, 236, 237, 239, 240, 242, 243, 244, 245, 246, 248, 248,
249, 250, 251, 251, 252, 253, 253, 253, 254, 254, 254, 254, 254, 254, 254, 254, 253, 253, 253, 252,
252, 251, 250, 250, 249, 248, 247, 246, 245, 243, 242, 241, 239, 238, 236, 235, 233, 231, 229,
227, 225, 224, 221, 219, 217, 215, 213, 210, 208, 206, 203, 201, 198, 195, 193, 190, 187, 185,
182, 179, 176, 173, 170, 167, 164, 161, 158, 155, 152, 149, 146, 143, 140, 137, 134, 131, 128,
125, 121, 118, 115, 112, 109, 106, 103, 100, 97, 94, 91, 88, 85, 82, 79, 76, 73, 71, 68, 65, 62,
60, 57, 55, 52, 50, 47, 45, 42, 40, 38, 36, 34, 31, 29, 27, 26, 24, 22, 20, 19, 17, 15, 14, 13,
11, 10, 9, 8, 7, 6, 5, 4, 3, 3, 2, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 3, 4, 4, 5, 6, 7,
8, 9, 10, 12, 13, 14, 16, 17, 19, 21, 22, 24, 26, 28, 30, 32, 34, 36, 39, 41, 43, 45, 48, 50, 53,
55, 58, 61, 63, 66, 69, 72, 74, 77, 80, 83, 86, 89, 92, 95, 98, 101, 104, 107, 110, 113, 116,
119, 122, };
```