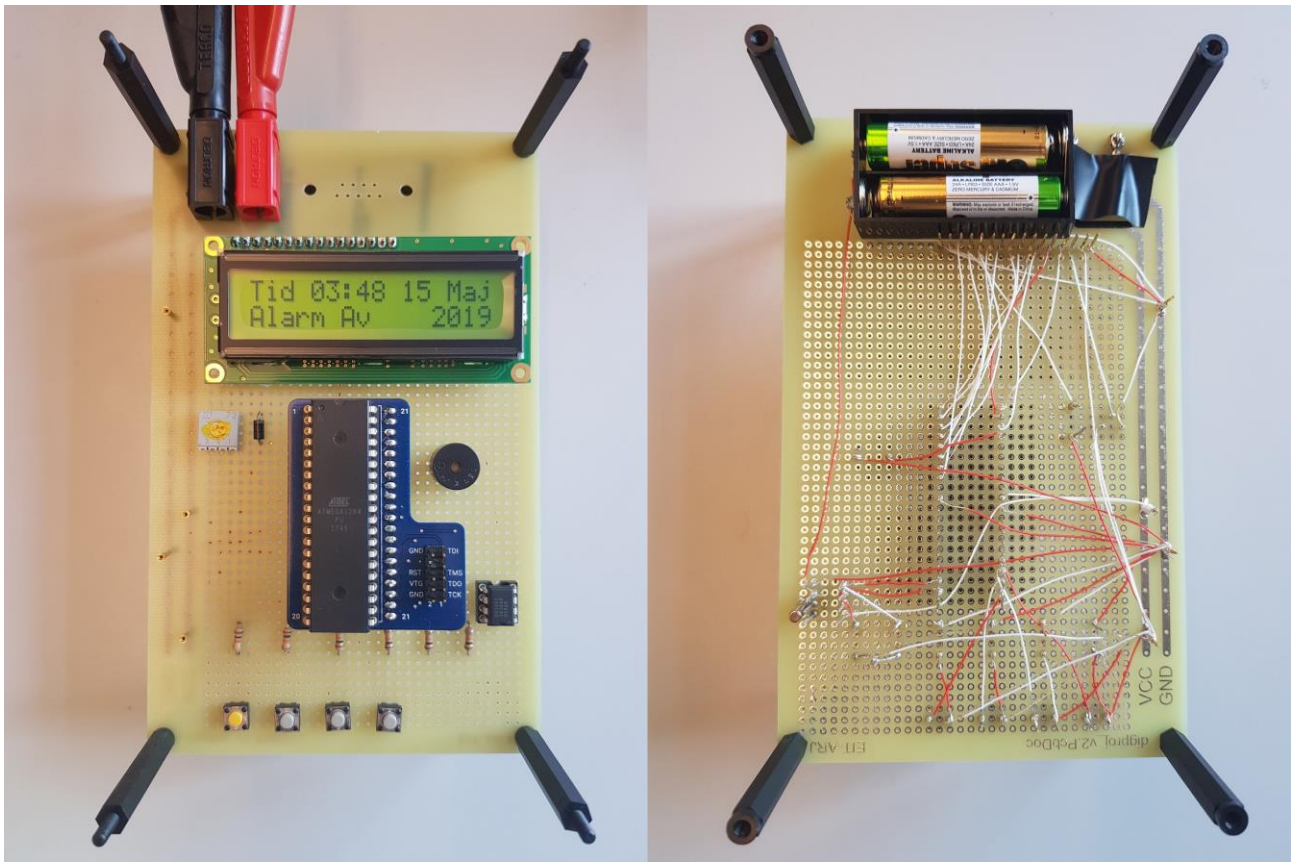
Lunds Tekniska Högskola

Digitala System Projekt - EITA15

VT - 2019

SUPER DUPER KLOCKAN



Handledare: Bertil Lindvall

Studenter :

NamWan Chansaeng

Janne Stojovski

Joakim Svensson

Daniel Löfgren

Abstract

The goal of this project has been to create an alarm clock from a programmable ATmega processor and components. The project was developed from a simple idea to become an actual prototype.
All together it's been an informative project and thought us how to use our knowledge in C-programing and computer science to create a digital prototype.

Innehållsförteckning

1. Sammanfattning

I denna rapport redogörs framställningen av Super Duper klockan. Med hjälp av en ATmega1284 processor, realtidsklocka, display och andra mindre komponenter framställdes en fungerande prototyp som visar tid, datum och alarm. Arbetet kom med många utmaningar och har varit väldigt lärorikt.

1.1 Inledning

Denna rapport redogör ett projekt som är en del av kursen Digitala System Projekt (EITA15) under läsperiod av VT19 som ges studenter vid Lunds Tekniska Högskola. Kursens mål är att ge studenter en inblick i hur ett konstruktionsarbete går till genom att låta studenterna bygga och testa sin egen prototyp baserad på mikroprocessorn ATmega1284.

I denna rapport beskrivs utveckling av Super Duper klockan. Klockan visar datum, tid och har en alarmfunktion som spelar upp en enkel melodi vid en förutbestämd tidpunkt. Det övergripande syftet med projektet är att bygga en elektronisk prototyp baserad på en mikroprocessor med hjälp av C-programmering, digitalteknik och datateknik.

2. Kravspecifikation

Syftet med kravspecifikationen är att bestämma vilka funktioner väckarklockan bör hantera samt uppfylla följande krav:

2.1 Funktioner

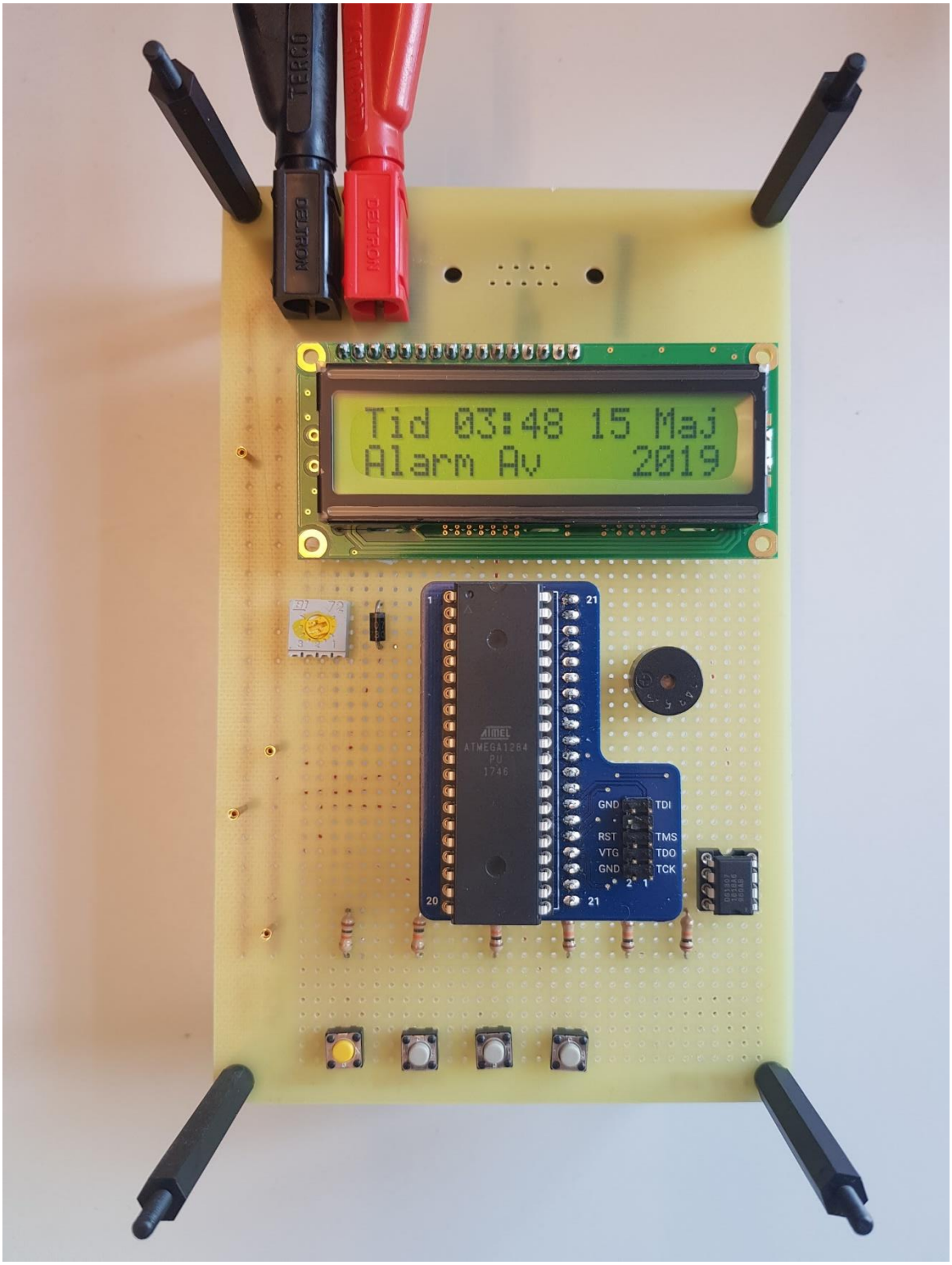
1. Displayen skall kunna visa datum som anges i formatet dag, månad, år.
2. Displayen skall kunna visa tid som anges i form av timmar (00-23) och minuter (00-59).
3. Displayen skall kunna visa alarm som anges i form av timmar (00-23) och minuter (00-59).
4. Det skall finnas en alarmfunktion.
5. Det ska framgå på displayen om larmet är aktiverat eller avaktiverat.
6. Larmet ska ge ifrån sig ljud med förutbestämd frekvens när det utlöses.

2.2 Funktionella krav

1. Det ska finnas en LCD-display för att möjliggöra kommunikation med användaren.
   Den ska visa datum, tid och alarmstatus.
2. Det ska finnas knapp 1 för att kunna slå på eller av alarmet.
3. Det ska finnas knapp 2 för att flytta pekaren åt vänster.
4. Det ska finnas knapp 3 för att flytta pekaren åt höger.
5. Det ska finnas knapp 4 för att ändra värdet på den position där pekaren sitter just nu..
6. Det ska finnas en summer som ger ifrån sig ljud.

3. Prototyp

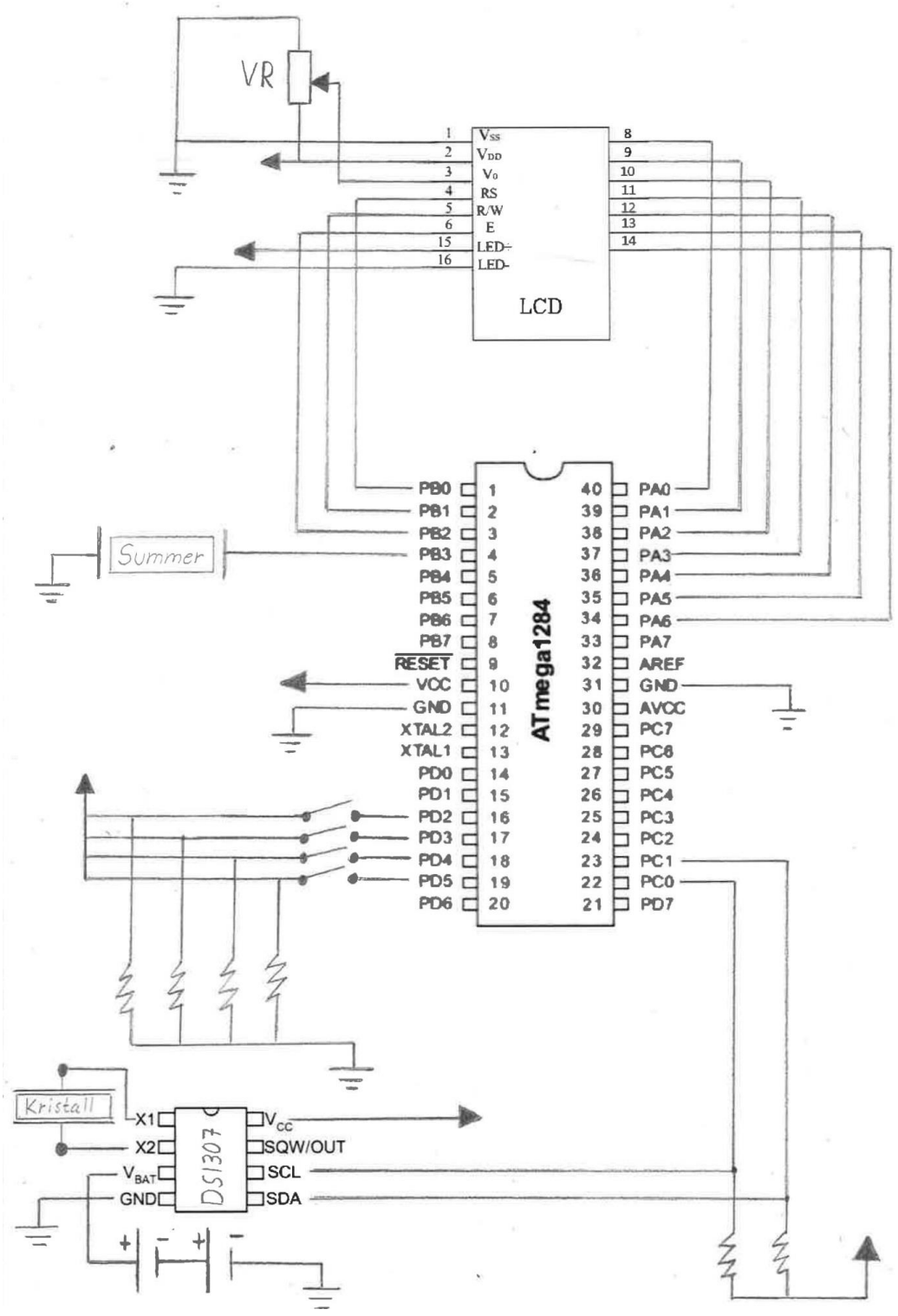
En prototyp på väckarklockan, Super Duper klockan.



Figur 1. Bild på Super Duper Klockan.

3.1 Kopplingsschema

Ett kopplingsschema som visar hur komponenterna är kopplade.



Figur 2. Kopplingsschemat för Super Duper Klockan.

4. Hårdvara

4.1 LCD - Display

Displayen som används är en SHARP Dot-Matrix, 2x16 radig. Displayen mottar kommando från processorn som beskriver vad som ska skrivas ut på skärmen i form av 8-bitar per tecken.

4.2 Processor

Processorn som används i det här projektet är en AVR AT Mega 1284 som består av 4 olika portar (A, B, C och D) och 40 pinnar. AT Mega 1284 programmeras i programmeringsspråket C med hjälp av programmet Atmel Studio 7.0 och vid överföring av programkod från dator till processorn används en JTAG.

4.3 Summer

Summern används för att ge ifrån sig ett ljud när alarmet utlöses. Ljudsignal genereras med en frekvens från processorn.

4.4 Realtidsklocka (RTC)

En Real-Time-Clock av modellen DS1307 används för att hålla koll på den aktuella tiden även om enheten stängs av eller det sker ett byte från vintertid till sommartid eller vice versa. Den 8-pin komponenten använder sig utav ett I^2C-gränssnittet.

4.5 Kristall

Kristallen används för att få en mer stabil och pålitlig frekvens vid beräkning av tiden. Kristallens frekvens är 32,768 KHz.

4.6 Knappar

För att göra det möjligt för användaren att ställa in tiden, alarmtiden och aktivera/avaktivera alarmet används fyra enkla knappar. Knapparna fungerar som en strömbrytare som sluter kretsen när de trycks ned och skickar en frekvens till processorn.

4.7 AVR JTAG

En AVR JTAG används för att överföra programkoden från dator till processorn.

4.8 Resistorer

Totalt används sex stycken resistorer av 10 kΩ. Fyra stycken resistorer till de fyra knappar och två resistorer till RTC med ett motstånd på 10 kΩ vardera.

4.9 Potentiometer

Potentiometern används för att reglera kontrasten på skärmen. Den är kopplad direkt till LCD-displayen.

4.10 Batterier

Två seriekopplade batterier på 1.5 Volt styck är kopplade till realtidsklockan för att även hålla reda på tiden när kretsen inte är inkopplad.

5. Mjukvara

Mjukvaran är programmerad i programmeringsspråket C med hjälp av programmet Atmel Studio 7.0

6. Arbetsprocessen

6.1 Planering och kravspecifikation

Projektet inleddes med en diskussion om vad som skulle konstrueras. Därefter skapades en kravspecifikation, en tidsplanering och en lista på alla komponenter som skulle användas. Utifrån komponenternas datablad konstruerades ett kopplingschema.

6.2 Konstruktion av hårdvaran

När kopplingsschemat hade godkänts av handledaren påbörjades bygget. Komponenterna fästes på kopplingsplattan och sladdarna mellan komponenterna pinnar virades fast enligt kopplingsschemat (se figur 2). När alla komponenter var på plats utfördes tester för att säkerställa så att hårdvaran var kopplad korrekt.

6.3 Programmering av mjukvaran

När alla tester var utförda påbörjade källkoden att skrivas i programspråket C. När koden var klar testades den med JTAG genom att se hur hårdvaran reagera vid olika anrop och kommandon.

6.4 Tillägg

Det har uppstått diverse problem under programmering och de flesta av problemen gick att lösa ganska fort genom att lägga till några komponenter som inte fanns med på det ursprungliga kopplingsschemat. De komponenterna som tillkom var en potentiometer för att kunna reglera kontrasten på displayen, och två seriekopplade batterier för att hålla reda på tiden när kretsen inte är inkopplad i PowerBoxen.

7. Resultat

Länk: https://www.youtube.com/watch?v=wBvdqDy96V4

8. Diskussion och slutsats.

Hela arbetet har varit intressant och framför allt väldigt lärorikt. Under projektets gång stötte vi på en del problem såsom att komponenter saknades men mestadels inom kodningen av mjukvaran. Det största problemet med kodningen var att få kommunikationen mellan RTC och I^2C att fungera. Men trots alla hinder lyckades vi konstruera en fungerande elektronisk prototyp med hjälp av våra kunskaper inom C-programmering, digitalteknik och datateknik.

9. Databladsreferenser

Kristall till klocka:
https://www.eit.lth.se/index.php?fsw=Datablad&dir=Periphery/Timers

Klocka:
https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Periphery/RTC/DS1307.pdf

Processor:
https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega1284.pdf

Summer:
https://www.elfa.se/sv/elektromagnetisk-summer-86-db-096-khz-vdc-kingstate-kxg1201b/p/13787249

Sharp DOT-MATRIX:
https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pd

11. Källkod

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

#define F_CPU 1600000UL;
#define SLA_W  0xD0; //Slav address + write bit (0)
#define SLA_Rd  0xD1; //Slav address + read bit (1)


uint8_t buttonSave;
char minutes;
char hour;
char date;
char month;
char year;
int alarmOn = 0;
char tidSec = 'A';
char tidSec10 = 'A';
char tidMin = 'A';
char tidMin10 = 'A';
char tidHour = 'A';
char tidHour10 = 'A';
char tidDat = 'A';
char tidDat10 = 'A';
char tidMon = 'A';
char tidMon10 = 'A';
char tidYear = 'A';
char tidYear10 = 'A';
char totMon = 'A';
char readTime;
char currentPos = 0;

char alarmSec = '1';
```

```
char alarmSec10 = '0';
char alarmMin = '5';
char alarmMin10 = '1';
char alarmHour = '8';
char alarmHour10 = '0';

char dupMin = 'A';
char dupMin10 = 'A';
char dupHour = 'A';
char dupHour10 = 'A';
char dupDat = 'A';
char dupDat10 = 'A';
char dupMon = 'A';
char dupMon10 = 'A';
char dupYear = 'A';
char dupYear10 = 'A';


char currentButton;
char previousButton;

int alarmToggled = 0;
char buttonResult;

void TWI_transmitTime(char address, char data);
char TWI_readTime(char adress);
void writeDisplay(char command);

/**
		FLYTTADE MAIN LÄNGST NER, PALLAR INTE DEKLARERA FUNKTIONER
**/

void timeInit(char y10, char y, char mon10, char mon, char dat10, char dat, char h10, char
h, char min10, char min, char sec10, char sec){

	char seconds = (sec10<<4)| sec;
	TWI_transmitTime(0x00, seconds);
	char minutes = (min10<<4)| min;
	TWI_transmitTime(0x01, minutes);
	char hour = (h10<<4)| h;
	TWI_transmitTime(0x02, hour);
	char date = (dat10<<4)| dat;
	TWI_transmitTime(0x04, date);
	char month = (mon10<<4)| mon;
	TWI_transmitTime(0x05, month);
	char year = (y10<<4)| y;
	TWI_transmitTime(0x06, year);
}

void timeOnDisplay(){

	//gör cursorn osynlig efteråt här

	if(tidMin10 != ((TWI_readTime(0x01)&0x70)>>4)){
		tidMin10 = ((TWI_readTime(0x01)&0x70)>>4);
		minutes = (tidMin10<<4)|tidMin;
		moveCursor(0x07);
		writeDisplay(tidMin10+48);
	}

	if(tidMin != (TWI_readTime(0x01)&0x0F)){
		tidMin = (TWI_readTime(0x01)&0x0F);
		minutes = (tidMin10<<4)|tidMin;
		moveCursor(0x08);
		writeDisplay(tidMin+48);
```

```
	}

	if(tidHour10 != (TWI_readTime(0x02)&0x30)>>4){
		tidHour10 = ((TWI_readTime(0x02)&0x30)>>4);
		hour = (tidHour10<<4) | tidHour;
		moveCursor(0x04);
		writeDisplay(tidHour10+48);
	}

	if(tidHour != (TWI_readTime(0x02)&0x0F)){
		tidHour = (TWI_readTime(0x02)&0x0F);
		hour = (tidHour10<<4) | tidHour;
		moveCursor(0x05);
		writeDisplay(tidHour+48);
	}

	if(tidDat10 != ((TWI_readTime(0x04)&0x30)>>4)){
		tidDat10 = ((TWI_readTime(0x04)&0x30)>>4);
		date = (tidDat10<<4) | tidDat;
		moveCursor(0x0A);
		writeDisplay(tidDat10+48);
	}

	if(tidDat != (TWI_readTime(0x04)&0x0F)){
		tidDat = (TWI_readTime(0x04)&0x0F);
		date = (tidDat10<<4) | tidDat;
		moveCursor(0x0B);
		writeDisplay(tidDat+48);
	}

	if(tidMon10 != ((TWI_readTime(0x05)&0x10)>>4)){
		tidMon10 = ((TWI_readTime(0x05)&0x10)>>4);
		month = (tidMon10<<4) | tidMon;
		writeMonth();
	}

	if(tidMon != (TWI_readTime(0x05)&0x0F)){
		tidMon = (TWI_readTime(0x05)&0x0F);
		month = (tidMon10<<4) | tidMon;
		writeMonth();
	}

	if(tidYear10 != ((TWI_readTime(0x06)&0xF0)>>4)){
		tidYear10 = ((TWI_readTime(0x06)&0xF0)>>4);
		year = (tidYear10<<4) | tidYear;
		moveCursor(0x4E);
		writeDisplay(tidYear10+48);
	}

	if(tidYear != (TWI_readTime(0x06)&0x0F)){
		tidYear = (TWI_readTime(0x06)&0x0F);
		year = (tidYear10<<4) | tidYear;
		moveCursor(0x4F);
		writeDisplay(tidYear+48);
	}

	moveCursor(currentPos);
}

void writeAlarmDisplay(){

	//detta är inte så effektivt, gör bara detta om alarmet har togglats av/på

	if(alarmOn==0){
		moveCursor(0x46);
```

```
                writeDisplay('A');
                writeDisplay('v');
                writeDisplay(' ');
                writeDisplay(' ');
                writeDisplay(' ');
                }else{
                moveCursor(0x46);
                writeDisplay(alarmHour10);
                writeDisplay(alarmHour);
                writeDisplay(':');
                writeDisplay(alarmMin10);
                writeDisplay(alarmMin);
        }
        moveCursor(currentPos);
}

void writeMonth(){
        totMon = tidMon + (tidMon10<<4);

        switch(totMon){
                case 0x01 :
                        moveCursor(0x0D);
                        writeDisplay('J');
                        writeDisplay('a');
                        writeDisplay('n');
                        break;

                case 0x02 :
                        moveCursor(0x0D);
                        writeDisplay('F');
                        writeDisplay('e');
                        writeDisplay('b');
                        break;

                case 0x03 :
                        moveCursor(0x0D);
                        writeDisplay('M');
                        writeDisplay('a');
                        writeDisplay('r');
                        break;

                case 0x04 :
                        moveCursor(0x0D);
                        writeDisplay('A');
                        writeDisplay('p');
                        writeDisplay('r');
                        break;

                case 0x05 :
                        moveCursor(0x0D);
                        writeDisplay('M');
                        writeDisplay('a');
                        writeDisplay('j');
                        break;

                case 0x06 :
                        moveCursor(0x0D);
                        writeDisplay('J');
                        writeDisplay('u');
                        writeDisplay('n');
                        break;

                case 0x07 :
                        moveCursor(0x0D);
                        writeDisplay('J');
```

```
                        writeDisplay('u');
                        writeDisplay('l');
                        break;

                case 0x08 :
                        moveCursor(0x0D);
                        writeDisplay('A');
                        writeDisplay('u');
                        writeDisplay('g');
                        break;

                case 0x09 :
                        moveCursor(0x0D);
                        writeDisplay('S');
                        writeDisplay('e');
                        writeDisplay('p');
                        break;

                case 0x10 :
                        moveCursor(0x0D);
                        writeDisplay('O');
                        writeDisplay('k');
                        writeDisplay('t');
                        break;

                case 0x11 :
                        moveCursor(0x0D);
                        writeDisplay('N');
                        writeDisplay('o');
                        writeDisplay('v');
                        break;

                case 0x12 :
                        moveCursor(0x0D);
                        writeDisplay('D');
                        writeDisplay('e');
                        writeDisplay('c');
                        break;

        }
}

void TWI_transmitTime(char address, char data){
        //https://www.eit.lth.se/fileadmin/eit/courses/eita15/avr-libc-user-manual-
2.0.0/util_2twi_8h.html
        TWCR |= (1 << TWINT) | (1 << TWSTA) | (1 << TWEN); //startar kommunikation
        while (!(TWCR & (1<<TWINT))){}; //väntar på interupptflaggan

        if((TWSR & 0xF8) != 0x08){ //kollar om start är skickat
                while(1){};
        }

        TWDR = SLA_W; //slave write address
        TWCR = (1<<TWINT) | (1<<TWEN); //startar process för att sända

        while (!(TWCR & (1<<TWINT))){}; //väntar på att twint flaggan ska bli hög

        if ((TWSR & 0xF8) != 0x18)  //kollar att sändarens write skickar ett ack
        while(1){};

        TWDR = address;                     //addressen vi vill skriva till
        TWCR = (1<<TWINT) | (1<<TWEN);

        while (!(TWCR & (1<<TWINT))){};
```

```
	if ((TWSR & 0xF8) != 0x28)		//kollar om datan är godkänd
	while(1){};

	TWDR = data;				//vad vi vill skriva till addressen
	TWCR = (1<<TWINT) | (1<<TWEN);

	while (!(TWCR & (1<<TWINT))){};

	if ((TWSR & 0xF8) != 0x28)
	while(1){};

	TWDR = 0;
	TWCR = (1<<TWINT) | (1<<TWEN);

	while (!(TWCR & (1<<TWINT))){};

	if ((TWSR & 0xF8) != 0x28)
	while(1){};

	TWCR = (1<<TWINT)| (1<<TWEN)| (1<<TWSTO); // stop
}

void TWI_crystalInit(){
	TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);
	while (!(TWCR & (1<<TWINT))){};

	if((TWSR & 0xF8) != 0x08){
		while(1){};
	}

	TWDR = SLA_W;
	TWCR = (1<<TWINT) | (1<<TWEN);

	while (!(TWCR & (1<<TWINT)));

	if ((TWSR & 0xF8) != 0x18)
	//https://www.eit.lth.se/fileadmin/eit/courses/eita15/avr-libc-user-manual-
2.0.0/util_2twi_8h.html
	while(1){};

	TWDR = 0x00;
	TWCR = (1<<TWINT) | (1<<TWEN);

	while (!(TWCR & (1<<TWINT))){};

	if ((TWSR & 0xF8) != 0x28)
	while(1){};

	TWDR = (0<<7);
	TWCR = (1<<TWINT) | (1<<TWEN);

	while (!(TWCR & (1<<TWINT))){};

	if ((TWSR & 0xF8) != 0x28)
	while(1){};

	TWCR = (1<<TWINT)| (1<<TWEN)| (1<<TWSTO);
}

char TWI_readTime(char address){			//börjar med att skriva till vilken
adress och sedan läser från adressen.
	TWCR |= (1 << TWINT) | (1 << TWSTA) | (1 << TWEN); //TWEN bit = 1, TWI enable. TWINT,
TWSKA bits = 1, start TWI create master.
	while (!(TWCR & (1<<TWINT))){};
```

```
	if((TWSR & 0xF8) != 0x08){
		while(1){};
	}

	TWDR = SLA_W;
	TWCR = (1<<TWINT) | (1<<TWEN);

	while (!(TWCR & (1<<TWINT))){};

	if ((TWSR & 0xF8) != 0x18)
	//https://www.eit.lth.se/fileadmin/eit/courses/eita15/avr-libc-user-manual-
2.0.0/util_2twi_8h.html
	while(1){};

	TWDR = address;
	TWCR = (1<<TWINT) | (1<<TWEN);

	while (!(TWCR & (1<<TWINT))){};

	if((TWSR & 0xF8) != 0x28){
		while(1){};
	}

	TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);
	while (!(TWCR & (1<<TWINT))){};

	if((TWSR & 0xF8) != 0x10){
		while(1){};
	}

	TWDR = SLA_Rd;
	TWCR = (1<<TWINT) | (1<<TWEN);

	while (!(TWCR & (1<<TWINT))){};

	if ((TWSR & 0xF8) != 0x40)
	while(1){};


	TWCR = (1<<TWINT) | (1<<TWEN);

	while (!(TWCR & (1<<TWINT))){};

	readTime = TWDR;

	if ((TWSR & 0xF8) != 0x58)
	while(1){};

	TWCR = (1<<TWINT)| (1<<TWEN)| (1<<TWSTO);
	return readTime;
}

void displayBusy(){ //oanvänd just nu
	DDRA=0x00;


	char data = PINA;
	char checkBusy=data & 0b10000000;
	while(checkBusy == 0b10000000){
		char bPorts = PORTB;
		PORTB = 0b00000010 | bPorts;
		PORTB = 0b00000110 | bPorts;

		checkBusy=PINA;
		checkBusy=PINA & 0b10000000;
```

```
	}

	DDRA=0xFF;
}

void moveCursor(char location){
	//0x40 eller 0x41 för andra raden
	commandDisplay(0b00010100);
	commandDisplay((1<<7) | location);
}

void Animation(){
	DDRA = 0xFF; //PA0-7 output
	DDRB = 0b00001111; //PB0-3 output (pb3 för summer)

	//commandDisplay(0x30); //function set
	commandDisplay(0b00111000); //function set
	commandDisplay(0b00001110); //display on/off control
	commandDisplay(0x01); //clear display


	moveCursor(0x0F);
	writeDisplay('S');
	writeDisplay('U');
	writeDisplay('P');
	writeDisplay('E');
	writeDisplay('R');

	moveCursor(0x55);

	writeDisplay('D');
	writeDisplay('U');
	writeDisplay('P');
	writeDisplay('E');
	writeDisplay('R');

	moveCursor(0x1B);

	writeDisplay('K');
	writeDisplay('L');
	writeDisplay('O');
	writeDisplay('C');
	writeDisplay('K');
	writeDisplay('A');
	writeDisplay('N');
	writeDisplay(' ');
	writeDisplay(':');
	writeDisplay(')');

	for(int j=0;j<19;j++){
		_delay_ms(200);
		commandDisplay(0b00011000);
		litetPip();
	}

}

void displayInit(){
	DDRA = 0xFF; //PA0-7 output
	DDRB = 0b00001111; //PB0-3 output (pb3 för summer)

	//commandDisplay(0x30); //function set
	commandDisplay(0b00111000); //function set
	commandDisplay(0b00001110); //display on/off control
	commandDisplay(0x01); //clear display
```

```
	writeDisplay('T'); //H
	writeDisplay('i');
	writeDisplay('d');
	writeDisplay(' ');
	//writeDisplay(0b10111100); //:)
	writeDisplay('0'); //minuter tiotal
	writeDisplay('7'); //minuter ental
	writeDisplay(':');
	writeDisplay('4'); //timmar tiotal
	writeDisplay('2'); //timmar ental
	writeDisplay(' ');
	writeDisplay(' '); //datum tiotal
	writeDisplay(' ');//datum ental
	writeDisplay(' ');
	writeDisplay('D');
	writeDisplay('e');
	writeDisplay('c');

	commandDisplay(0b00010100); //flytta cursorn ett steg till höger, behövs för att det
nedanför ska funka
	moveCursor(0x40); //hoppa till andra raden
	writeDisplay('A');
	writeDisplay('l');
	writeDisplay('a');
	writeDisplay('r');
	writeDisplay('m');
	writeDisplay(' ');
	writeDisplay('A');
	writeDisplay('v');
	writeDisplay(' ');
	writeDisplay(' ');
	writeDisplay(' ');
	writeDisplay(' ');
	writeDisplay('2');
	writeDisplay('0');
	writeDisplay('1');
	writeDisplay('9');
}

void writeDisplay(char command){
	monitorDelay();
	PORTB  = 0b00000101;
	PORTA  = command;
	PORTB  = 0b00000000;

}

void commandDisplay(char command){
	PORTB=0b00000100;
	PORTA=command;
	PORTB=0b00000000;
	monitorDelay();
	PORTB=0b00000100;
}

void monitorDelay(){
	_delay_ms(2);
}

void cursorMovement(){

	//if alarmPå == 1, hoppa till 0x46 -> skriv 11:11
```

```
	if(currentPos==0x4F){
		currentPos = 0;
		commandDisplay(0x0C);
	}

	else if(currentPos==0){
		commandDisplay(0x0D);
		currentPos = 0x04;
	}

	else if(currentPos==0x04){
		currentPos += 1;
	}

	else if(currentPos==0x05){
		currentPos = 0x07;
	}

	else if(currentPos==0x07){
		currentPos += 1;
	}

	else if(currentPos==0x08){
		currentPos = 0x0A;
	}

	else if(currentPos==0x0A){
		currentPos += 1;
	}

	else if(currentPos==0x0B){
		currentPos = 0x0D;
	}

	else if(currentPos==0x0D){
		if(alarmOn==1)
		currentPos = 0x46;
		else
		currentPos = 0x4E;
	}
	//OM ALARM ÄR PÅ
	else if(currentPos == 0x46){
		currentPos = 0x47;
	}
	else if(currentPos == 0x47){
		currentPos = 0x49;
	}
	else if(currentPos == 0x49){
		currentPos = 0x4A;
	}
	else if(currentPos == 0x4A){
		currentPos = 0x4E;
	}

	else if(currentPos == 0x4E){
		currentPos = 0x4F;
	}

}

void reverseMovement(){

	//if alarmPå == 1, hoppa till 0x46 -> skriv 11:11
```

```
	if(currentPos==0){
		currentPos = 0x4F;
		commandDisplay(0x0D);
	}

	else if(currentPos==0x04){
		currentPos = 0;
		commandDisplay(0x0C);
	}

	else if(currentPos==0x05){
		currentPos = 0x04;
	}

	else if(currentPos==0x07){
		currentPos = 0x05;
	}

	else if(currentPos==0x08){
		currentPos = 0x07;
	}

	else if(currentPos==0x0A){
		currentPos = 0x08;
	}

	else if(currentPos==0x0B){
		currentPos = 0x0A;
	}

	else if(currentPos==0x0D){

		currentPos = 0x0B;
	}
	//OM ALARMET ÄR PÅ
	else if(currentPos == 0x4E){
		if(alarmOn==1)
		currentPos = 0x4A;
		else
		currentPos = 0x0D;
	}
	else if(currentPos==0x4A){

		currentPos = 0x49;
	}
	else if(currentPos==0x49){

		currentPos = 0x47;
	}
	else if(currentPos==0x47){

		currentPos = 0x46;
	}
	else if(currentPos==0x46){

		currentPos = 0x0D;
	}

	else if(currentPos==0x4F){
		currentPos = 0x4E;
	}

}

void alarmTrig(){
```

```
	if(alarmHour==tidHour+48 && alarmHour10 == tidHour10+48 && alarmMin10 == tidMin10+48
&& alarmMin == tidMin+48){
		if(alarmOn==1){
		litetPip();
		}
	}
}

void testCommit(){
	TWI_transmitTime(0x01, minutes);
	TWI_transmitTime(0x02, hour);
	TWI_transmitTime(0x04, date);
	TWI_transmitTime(0x05, month);
	TWI_transmitTime(0x06, year);
}

void changeTime(){
	//deklarera nya variabler för tidMin osv.., i början av alla case satser p.g.a att vi
använder den andra för att kolla uppdateringen.
	switch(currentPos){
		case 0x07 :
		dupMin10 = tidMin10;
		if(tidMin10 == 5){
			dupMin10 = 0;
			} else{
			dupMin10++;
		}
		minutes = (dupMin10<<4)| tidMin;
		break;
		case 0x08 :
		dupMin = tidMin;
		if(dupMin == 9){
			dupMin = 0;
		}
		else{
			dupMin++;
		}
		minutes = (tidMin10<<4)| dupMin;
		break;
		case 0x04 :
		dupHour10 = tidHour10;
		if(tidHour10 == 2){
			dupHour10 = 0;
		}
		else if(tidHour10==1 && tidHour>=4){
			dupHour10 = 0;
		}
		else{
			dupHour10++;
		}
		hour = (dupHour10<<4)| tidHour;
		break;
		case 0x05 :
		dupHour = tidHour;
		if(tidHour == 9 && tidHour10<2){
			dupHour = 0;
		}
		else if(tidHour10 == 2 && tidHour >= 4){  // i mån av tid lägger vi till + 10 h
			dupHour = 0;
		}
		else if(tidHour==9){
			dupHour++;
		}
		else{
			dupHour++;
```

```
		}
		hour = (tidHour10<<4)| dupHour;
		break;
		case 0x0A :
		dupDat10 = tidDat10;
		if(tidDat10 == 3){
			dupDat10 = 0;
		}
		else if(tidDat10==2 && tidMon==2 && tidMon10==0){
			dupDat10 = 0;
		}
		else if (tidDat10==2 && tidDat >= 2){
			dupDat10 = 0;
		}
		else{
			dupDat10++;
		}
		date = (dupDat10<<4)| tidDat;
		break;
		case 0x0B :
		dupDat = tidDat;
		if(tidDat == 8 && tidMon10 == 0 && tidMon == 2 && ((tidYear+(tidYear10))%4)!=
0){ //undantag feb ska fixas
			dupDat = 0;
		}
		else if(tidDat == 9 && tidDat10 >= 1){
			dupDat = 0;
		}
		else if(tidDat == 9){
			dupDat = 1;
		}
		else if (tidDat>=1&&tidDat10==3){
			dupDat = 0;
		}
		else{
			dupDat++;
		}
		date = (tidDat10<<4)| dupDat;
		break;
		case 0x0D :
		dupMon10 = tidMon10;
		dupMon = tidMon;
		if(tidMon == 9 && tidMon10==0){
			dupMon = 0;
			dupMon10 = 1;
		} else if(dupMon10 == 1 && dupMon == 2){
			dupMon = 1;
			dupMon10 = 0;
		}
		else {
			dupMon++;
		}
		month = (dupMon10<<4)| dupMon;
		break;
		case 0x4F :
		dupYear = tidYear;
		if(tidYear == 9){
			dupYear = 0;
		}
		else{
			dupYear++;
		}
		year = (tidYear10<<4)| dupYear;
		break;
```

```
		case 0x4E :
		dupYear10 = tidYear10;
		if(tidYear10 == 9){
			dupYear10 = 0;
		}
		else{
			dupYear10++;
		}
		year = (dupYear10<<4)| tidYear;
		break;

		case 0x49 :
		if(alarmMin10 == 53){
			alarmMin10 = 48;
			} else{
			alarmMin10++;
		}
		break;
		case 0x4A :
		if(alarmMin == 57){
			alarmMin = 48;
		}
		else{
			alarmMin++;
		}
		break;
		case 0x46 :
		if(alarmHour10 >= 50){
			alarmHour10 = 48;
		}
		else if(alarmHour10==49 && alarmHour>=52){
			alarmHour10 = 48;
		}
		else{
			alarmHour10++;
		}
		break;
		case 0x47 :
		if(alarmHour == 57 && alarmHour10<50){
			alarmHour = 48;
		}
		else if(alarmHour10 >= 50 && alarmHour >= 52){  // i mån av tid lägger vi till
+ 10 h
			alarmHour = 48;
		}
		else{
			alarmHour++;
		}

		break;
	}

}

void litetPip(){
	for(int i=0;i<3;i++){
		_delay_ms(1);
		PORTB = (1<<3);
		_delay_ms(1);
		PORTB = (0<<3);
	}
}

void Alarm(){
	for(int i=0;i<20;i++){
```

```
			litetPip();
		}
}

void alarmOnOff(){
		if(currentPos == 0x46 ||  currentPos ==0x47 ||  currentPos ==0x49 ||  currentPos
==0x4A){
			currentPos = 0x04;
		}

		if(alarmOn==1){
			alarmOn=0;
		}
		else if(alarmOn==0){
			alarmOn=1;
		}

		writeAlarmDisplay();
}


char buttonDebounce(){
		previousButton = PIND;
		_delay_ms(20);
		currentButton = PIND;
		if(currentButton == previousButton){
			_delay_ms(100);
			return currentButton;
			}else{
			return 0;
		}
		return 0;
}

void resetPORTD(){
		DDRD = 0xFF;
		PORTD = 0xFF;
		_delay_ms(1);
		PORTD = 0x00;
		DDRD = 0x00;
}

void buttonAction(char Button){

		if(buttonResult != 0){
			switch(buttonResult) {
				case 0x04  :
				alarmOnOff();
				break;

				case 0x08  :
				reverseMovement();
				moveCursor(currentPos);
				break;

				case 0x10  :
				cursorMovement();
				moveCursor(currentPos);

				break;

				case 0x20  :
				changeTime();
				writeAlarmDisplay();
				break;
```

```
                }
        }

}

int main(void){
        DDRD = 0x00; //Knappar insignal
        //Alarm();
        //TWI_crystalInit();
        //timeInit(1, 9, 0, 5, 1, 6, 1, 4, 0, 2, 5, 0);
        Animation();
        displayInit();

        while (1){
                timeOnDisplay();
                alarmTrig();

                testCommit();

                resetPORTD();
                buttonResult = buttonDebounce();
                buttonAction(buttonResult);
        }
}
```