LUNDS TEKNISKA HÖGSKOLA
Lunds universitet

Institutionen för Elektronik och Informationsteknik
Digitala Projekt

Ljuskänslig riktningsautomatik

Styrt med MC68008

Handledare: Bertil Lindvall

DigP04
Anders Larsson 840125-3979
Mattias Jönsson 811126-4076

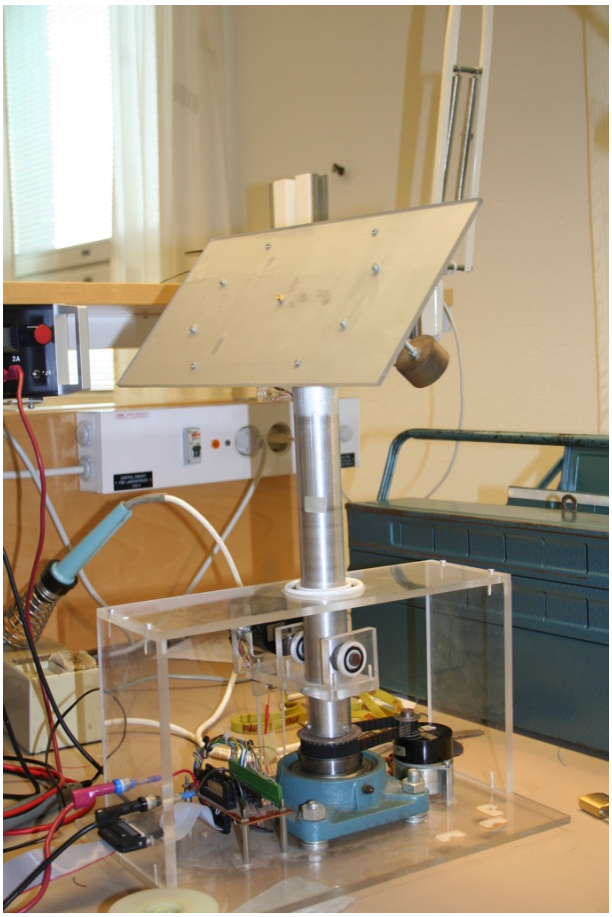
Abstract

This report discuss a digital project where a MC68008 processor with environment is implemented to control a lighttracking device. The device is built with two stepmotors for control in x and y directions. Four phototransistors are mounted as a detector on the device. The phototransistors are A/D converted in order to determine in which directions the stepmotors will act to move towards the light. Included in the project is to implement the processor with memories, control logic and I/O's with necessary A/D converting.

Innehåll

1 Inledning

Digitala projekt är en projekt med syfte att ge förståelse för hur ett datorsystem är uppbyggt. I kursen ingår momenten konstruktion med testning samt rapportskrivning och redovisning. Kursen är väldigt praktiskt orienterad och tyngdpunkten ligger i att bygga en fungerande konstruktion. Denna rapport behandlar konstruktionen av en rigg som följer en ljuskälla. Riggen detekterar ljus med hjälp fototransistorer och ställer in sig med hjälp av stegmotorer. För bild på rigg se figur 1. Rapporten är indelad i kapitlen *Metod*, *Resultat* samt *Slutsatser och Diskussion*



Figur 1: Ljussökande rigg med fotottransistorer och stegmotorer.

2 Metod

Detta kapitel behandlar uppsatt kravspecifikation och utvecklingsarbetet mot färdig implementation. Arbetsgången var att först konstruera fungerande hårdvara så långt detta var möjligt och sedan påbörja mjukvaruutvecklingen.

2.1 Kravspecifikation och grundläggande uppbyggnad

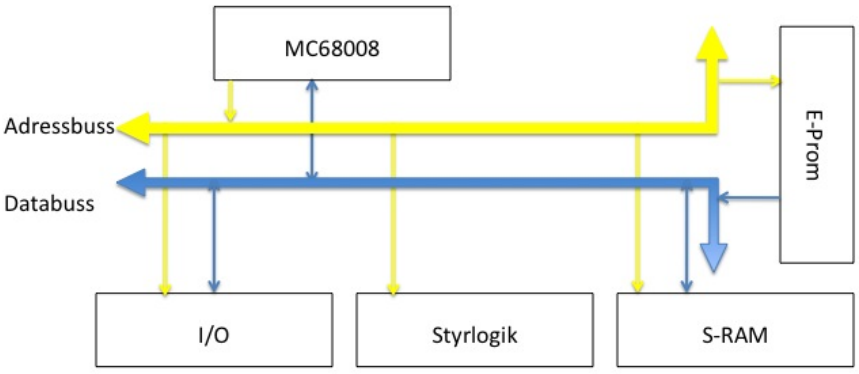
Kravspecifiktionen var det första steget i konstruktionen, denna styrde vad den slutgiltiga implementationen skulle behärska.

2.1.1 Kravspecifikation

Inledningsvis ställs en kravspecifikation upp, i detta fall:

1. Ta in 5 analoga signaler från fototransistorerna.
2. Omvandla dessa till digitala signaler.
3. Använda dessa signaler till att avgöra i vilken riktning ljuskällan befinner sig.
4. Avgöra i vilken riktning samt vilken motor som ska agera.
5. Generera en styrsignal till aktuell motor.
6. Skicka signal till motor.
7. Starta ny läsning.

Fototransistorerna avgör riktningen på ljuskällan. Dessa ger en analog signal som A/D-omvandlas och läses in i minnet. Dessa signaler behandlas av processorn och genom skrivna algoritmer avgörs hur processorn ska agera. Med detta bestämt skickas styrsignaler till motorerna.



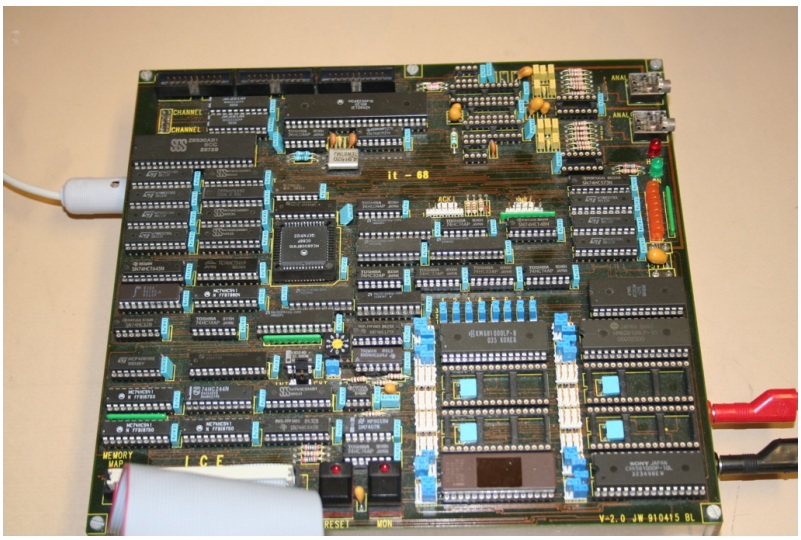
Figur 2: Principskiss över datorsystemet.

2.1.2 Grundkonstruktion, uppbyggnad

Grunden i konstruktionen är Motorola-processorn MC68008. Denna processor skall vara ansluten till ett programminne, arbetsminne, styrlogik och I/O-enheter se figur 2. Detta då MC68008 är en renodlad processor och saknar dessa enheter, extern klockning är även nödvändigt. Processorn samverkar med enheterna via adressering på 20-bitars adressbuss och dataöverföring på 8-bitars databuss.

2.1.3 Utvecklingsmiljö och stegvis testning

Då M68008 är väldigt enkel i sitt utförande krävs att användaren bygger miljö till processorn. För att detta ska kunna ske med kontinuerlig, stegvis testning används it-68 utvecklingssystem. Detta är ett system som är utvecklat på LTH av Bertil Lindvall. För bild på testtrustning se figur 3.



Figur 3: Testmiljön it68 ansluts till kretskortet i processorns sockel.

2.1.4 Mjukvara

Enligt uppsatta krav skall riggen klara att helt autonomt kunna följa en ljuskälla. För att styra hårdvaran krävs algoritmer för riktningsberäkning och stegmotorstyrning.

2.2 Konstruktion av Hårdvara

Stor del av projektet består i att designa en fungerande hårdvara. Detta kapitel behandlar de hårdvarukomponenter som är nödvändiga för en fungerande konstruktion.

2.2.1 Styr- och avbrottslogik

För att kontrollera trafik på databussen krävs att processorn styr vilka enheter som ska läsa respektive skriva på bussen. Detta sker genom att 3 bitar av adressbussen går till styrlogik som skickar styrsignaler till alla enheter anslutna till databussen.

Då avbrott är ett effektivt sätt att låta enheter arbeta cykliskt och sedan bryta huvudprogrammet då information finnes skulle detta implementeras i hårdvarukonstruktionen. MC68008 har två avbrottsnivåer som kan utnyttjas av användaren. För att kontrollera när avbrotten ska ske konstrueras även en avbrottslogik.

För att kombinera insignaler till fungerande styr- och avbrottslogik framgick det tidigt att programmerbara logikkretsar var att föredra. För att konstruera styrlogik för detta system krävdes sex insignaler samt tio utsignaler. För att konstruera avbrottslogik krävdes fem insignaler samt fyra utsignaler. Då dessa logiknät är till stordel oberoende av varandra insågs snabbt fördelen med att använda två separata programmerbara logikkretsar.

2.2.2 Programminne

Programminne, i detta fallet EPROM är nödvändigt för att lagra instruktioner till processorn. Det är i EPROMet som mjukvaran till systemet lagras och läses från.

2.2.3 Arbetsminne

Arbetsminne, i denna konstruktion SRAM används eftersom ingen skrivning till EPROM är möjlig. Detta ger att alla variabler och indata intitieras i SRAM. SRAM nollställs vid spänningsfall.

2.2.4 Klockkrets

Då MC68008 saknar intern klocka är det nödvändigt att klocka processorn externt. Detta löses med hjälp av en klockkrets uppbyggd kring en kristall. Denna krets består av kristall, kondensatorer, resistorer samt krets med två opförstärkare. Klockkretsen klockade även logikkretsar, samt nerdelat med D-vippa klockades även A/D-omvandlaren.

2.2.5 Input/Output

För att kunna ta in data till behandling samt skicka ut signaler krävs åtta utgångskanaler till stegmotorerna samt sju ingångskanaler till sensorer och knappar.

Då in- och utsignaler är parallella åttabitars signaler valdes det att använda två separata kretser. En för in- och en för utsignaler.

2.2.6 Analog till Digitalomvandling

Signalerna från fototransistorerna är en ström varierande på vilken ljusstyrka fototransistorn är utsatt för, behöver denna omvandlas till spänning som sedan A/D-omvandlas för att kunna nivåbestämmas digitalt.

A/D-omvandlingen skall hantera fyra separata kanaler och lägga ut resultatet på databussen, då databussen är på åtta bitar valdes en A/D-omvandlare med denna upplösning. Detta motsvarar digitala nivåer på utsignalen från 0-255. Motstånd valdes så att maxnivån lades på $5v$, och 0-nivån lades på $0v$.

2.2.7 Drivsteg till stegmotorer

Stegmotorerna styrs med vardera fyra transistorer vilka varieras i tiden för att röra motorn. Då signaler internt inom datorsystem har strömmar i storleksordningen μA krävs ett drivsteg för att för att kunna leverera de basströmmar som transistorerna kräver, i storleksordningen mA.

Styrsignalerna till stegmotorerna levereras från utgående parallellport med åtta bitar, fyra till varje motor. För att förstärka dessa innan de går in i riggen valdes ett åttakanals drivsteg.

2.2.8 Knappar och Ändlägessensorer

Förutom resetknapp till processorn behövs även ett reservsystem med knappar för att styra riggen manuellt . Detta fyller en stor funktion vid testning av stegmotordrivning när denna programmeras. Signalerna från knapparna till stegmotorstyrningen går genom ingående parallellport in på databussen.

På samma inport är även två stycken ändlägessensorer anslutna, dessa för att inte plattan med fototransistorerna skall nå sina ändlägen. Dessa sensorer genererar en $5v$ signal då de detekterar ändläge.

2.3 Konstruktion av mjukvara

Mjukvaran för systemet skrevs i Ansi C, detta är en standardversion av C. Upplägget vid konstruktion var att merparten av instruktioner skulle utföras av funktioner eller underprogram.

2.3.1 Avbrottsprogram exp2 och exp5

`exp5` är ett cyklisk avbrott i som aktiveras varje $0{,}1ms$, i fortsättningen benämt som tic. Vid aktivering kontrolleras om någon knapp är nedtryckt alternativt om något ändläge detekterat. Om någon knapp är nedtryckt ges instruktion till aktuell stegmotor. Vid utslag från ändlägesgivare stoppas vidare rörelse mot ändläget, efter detta kvitteras avbrottet. Vid varje 25e tic startas A/D-omvandlaren omvandling.

Avbrottsrutinen `exp2` som är ett avbrott av lägre grad än `exp5` aktiveras när A/D-omvandlaren skickar signal på att omvandling skett. Då aktiveras program för avläsning och sedan kvitteras detta avbrott.

2.3.2 A/D-omvandling

Funktioner för A/D-omvandlaren är:

```
aadstart()
aadread()
```

`aadstart()` är en funktion för att starta A/D-omvandlaren, denna anropas från `exp5` enligt ovan. Funktionen cyklar kanalerna från ch0 till ch3, detta görs med en räknare i funktionen.

`aadread()` är funktion för att läsa värde på A/D-omvandlaren, funktionen anropas från `exp2` då omvandling skett. Samma variabel som används i aadread används för att veta vilken kanal som omvandlats. Alla värden som läses in sparas i en vektor för aktuell kanal. Vektorn innehåller de senaste fyra värden som lästs från aktuell kanal. Vektorn medelvärdesbildas även efter varje läsning och på så vis uppdateras en ytterligare vektor som innehåller medelvärden från de fyra kanalerna. Vektorn med medelvärdena används vid beräkning av ljuskällans riktning

2.3.3 Stegmotorstyrning

För att generera sekvenser för stegmotorstyrning behövdes antalet steg i varje cykel bestämmas, då den enklaste cykel bestod av fyra bitmönster och denna fungerade bra valdes detta. Naturligt delades instruktionerna för styrning upp i funktionerna:

```
up(nbr)
down(nbr)
clockw(nbr)
counterclockw(nbr)
```

Dessa funktioner generar nbr antal cykler i given riktning, genom att lägga ut sekvenserna på databussen.

2.3.4 Riktningsbestämning

För att riggen ska kunna följa ljuskällan krävs en koppling mellan inlästa värden och motorstyrning. För att lösa detta skapades två funktioner som agerar utifrån värden på fototransistorerna:

`autox()`
`autoy()`

`autox()` kontrollerar i vilken x-led ljuset är starkast och anropar `clockw()` och `counterclockw()` utefter detta. Då riktningarna blir omvända då plattan passerat sitt mittläge finns även en räknare i yled som kontrollerar detta och spegelvänder kommandona vid behov.

`autoy()` är en funktion som styr riggen i y-led efter i vilken riktning ljuset är starkast, detta genom att anropa funktionerna `up()` och `down()`.

2.3.5 Huvudprogram

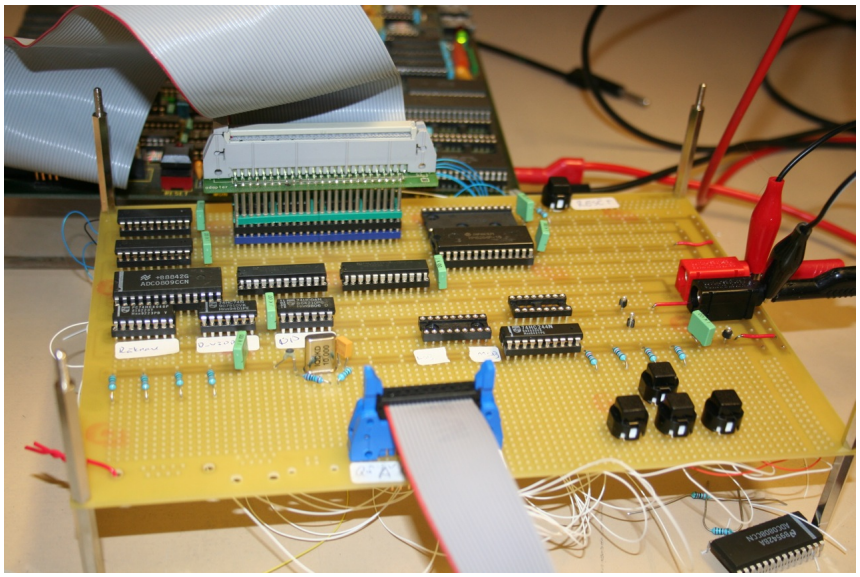
För att knyta samman alla funktioner skapas ett huvudprogram:

`run()`

Huvudprogrammet `run()` anropas vid uppstart och innehåller en oändlig loop som sköter beräkningen av i vilken riktning riggen skall korrigeras och anropar då `autox()` alternativt `autoy()`.

3 Resultat

Detta kapitel behandlar resultatet av utvecklingsarbetet. Här presenteras slutgiltig konstruktion av kretskort samt tillhörande programvara.



Figur 4: Hålkort med monterade kretsar, tesmiljön ansluten istället för MC68008.

3.1 Hårdvarukonstruktion

All styrning av adressbussar och databussar styrs från en programmerbar logikkrets benämnd med CSLOGIC i kretsschemat. Avbrottsstyrning till processorn sker genom programmerbara kretsen benämnd AVBLOGIC i kretsschemat. Styr- och avbrottslogik programmerades enligt Appendix C. Med logiken implementerad tillsammans med enheter nämnda i avsnittet *metod*

var en fungerande hårdvarukonstruktion byggd. Motorstyrningen implementerades med fyra steg i varje sekvens, detta var fullt tillräckligt för denna applikation, styrsekvenserna implementerades enligt figur 5. För slutigiltig hårdvarukonstruktionen se Appendix A. Efter ritning över systemet var färdig implementerades lösningen på ett hålkort med socklar och virning, se figur 4.

Steg	Q_1	Q_2	Q_3	Q_4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	1	0
1	1	0	1	0

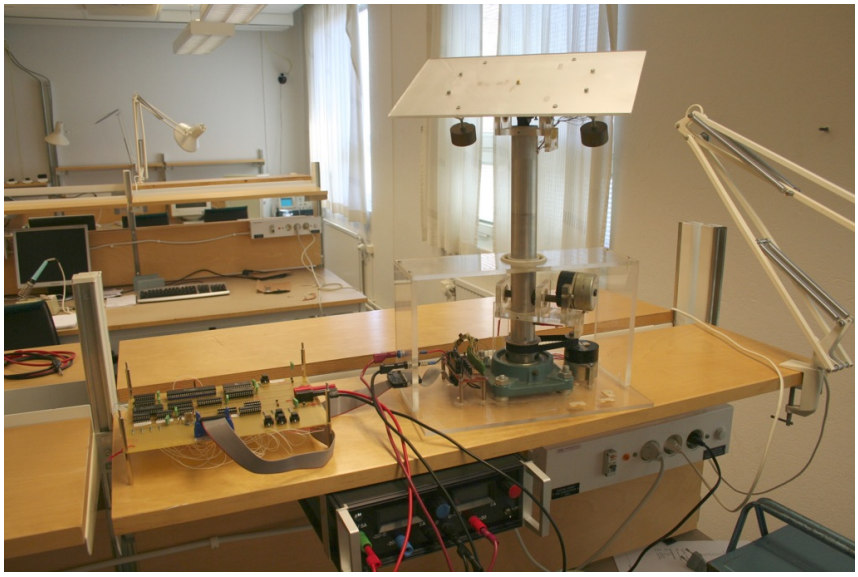
Figur 5: Styrsignaler för en stegmotorcykel

3.2 Mjukvarukonstruktion

Den färdiga mjukvaran kompilerades till maskinkod för att laddas ner till EPROM. När mjukvaran skall anpassas för EPROM är det nödvändigt att alla variabler och intieringar görs i huvudprogrammet, detta för att adresser till SRAM skall skapas samt att adresserad minnesplats i SRAM tilldelas ett värde. För komplett Ckod se Appendix B.

3.3 Systemkonstruktion

Det slutgiltiga systemet följer uppsatt kravspecifikation. Systemet arbetar autonomt med startvillkoret att riggen befinner sig i arbetsområdet mellan ändlägena. För bild över slutgiltigt system se figur 6.



Figur 6: Systemet arbetar självständigt och följer ljuskällan.

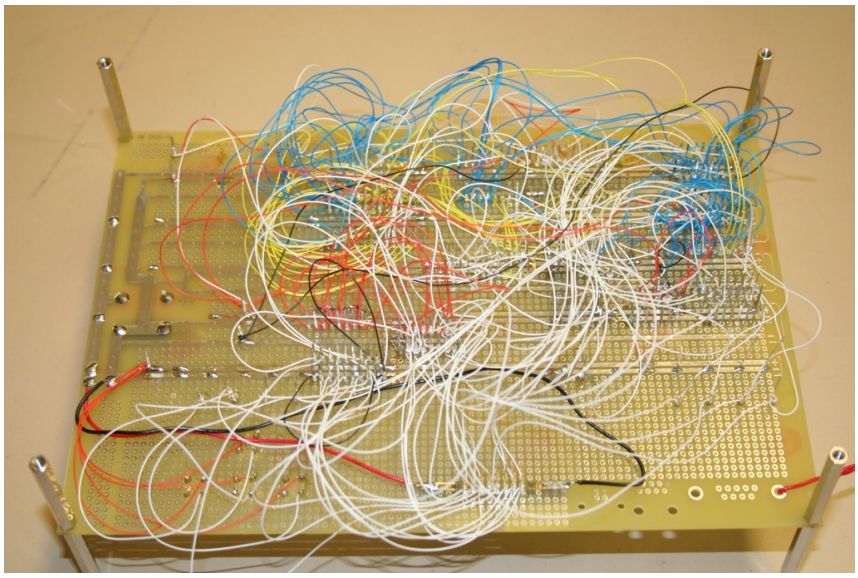
4 Slutsatser och Diskussion

Då ett datorsystem skall implenteras med med I/O och periferienheter kommer problem längs vägen att uppstå. Lärdomarna har varit många, detta kapitel tar upp våra större bekymmer och lärdomar av dessa.

4.1 Lärdomar och problem

En lärdom som dragits är att konstruktion av även ett enkelt system som detta kan generera många problem och bekymmer under projektets gång. Vikten av kontinuerlig testning har också visat sig vara av stor betydelse, detta då vi kört fast ordentligt på några punkter. Några bekymmer längs vägen har varit implementering av styrlogik, motorstyrning och adressera rätt kanal på A/D-omvandlaren.

Styrlogiken hade vi en del bekymmer med i början, detta var mest ett resultat av ovana och även några enkla misstag. Vi visste inte exakt vilka signaler vi skulle generera och när. Dessa bekymmer åtgärdades ganska enkelt och sedan fungerade denna biten utan bekymmer, vilket var väldigt uppskattat då antalet ledningar på undersidan av kortet växte snabbt. Se figur 7. Vad gällde motorstyrningen var här ett betydligt större problem, vi fick ej signaler



Figur 7: Antalet ledare på undersidan av kortet växte snabbt.

stabila då riggen var inkopplad. Signalerna var stabila då kretskortet kördes men då riggen var inkopplad var resultatet klart slumpartat. Efter ca två veckors arbete med darlington drivsteg med mera helt utan framgång implementerades en befintlig lösning med ett 74HC244, en åttakanals buffert. Syftet med bufferten är enbart att hålla signalen stabil och samtidigt ge transistorerna tillräckligt med ström för att reagera.

A/D-omvandlaren var nästa stora bekymmer, denna verkade till en början fungera bra men då testning med mjukvara inleddes märktes att den enbart läste från kanal 0. Detta problem diskuterades med handledaren under en vecka och vi kunde ej hitta någon lösning till varför addresseringen av kanalerna ej gick fram. Som ett sista desperat försök testades att ändra signalen för stabil adress från processorn(AS) till en signal som håller längre, nämligen den för stabil data(DS). Med denna ändring fungerade addresseringen till A/D-omvandlaren utmärkt,

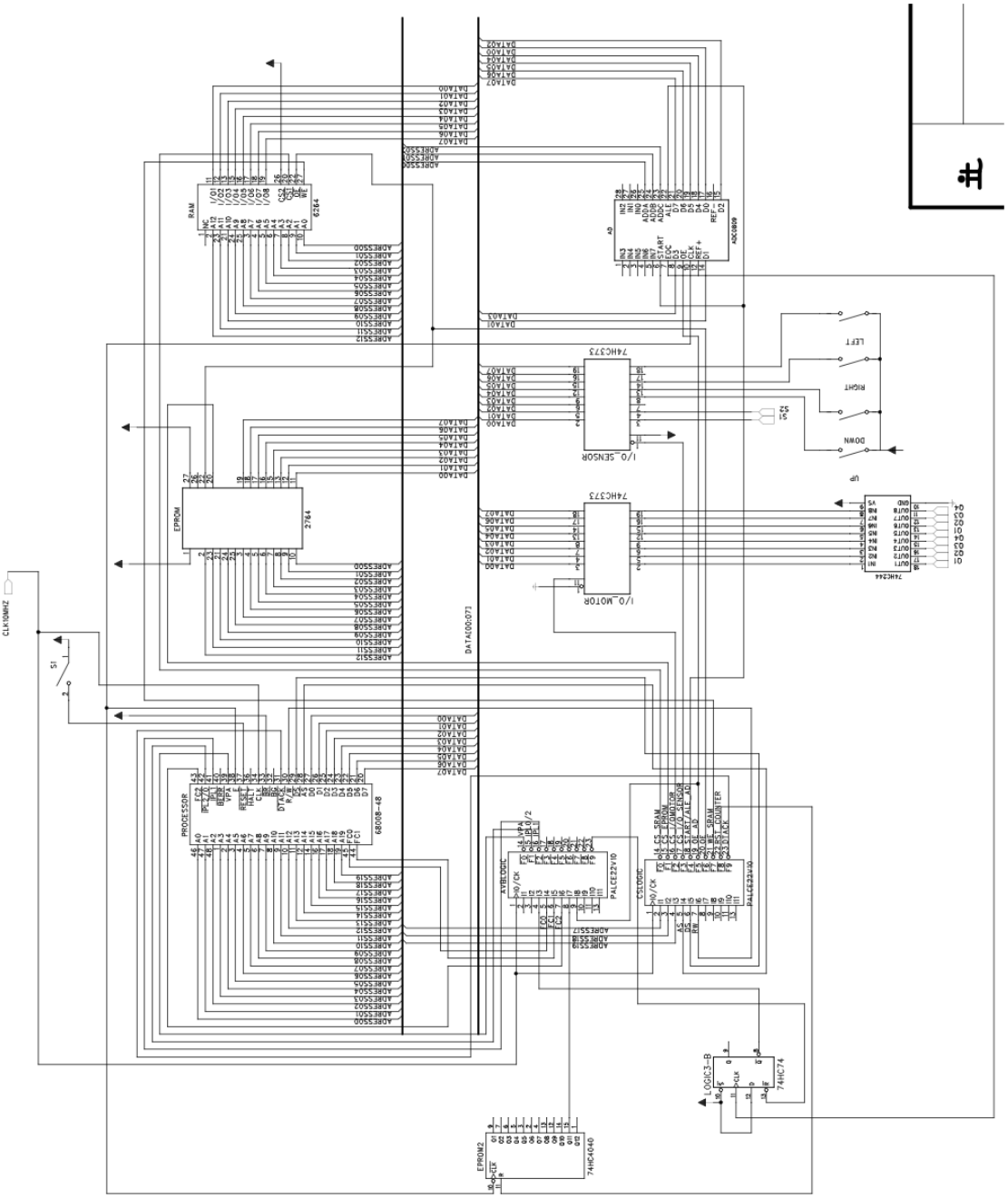
enda förklaringen till detta är att DS är en längre signal. Signalen AS borde dock fungerat till addresseringen och varken vi eller handledaren fann någon lösning till varför det ej gjorde det.

Med dessa problem avklarade var det enkelt att testa mjukvaran kontinuerligt på riggen. Utvecklingen av mjukvara begränsades i våra kunskaper av C, men detta var aldrig några större problem då programmet ej behövde bestå av några avancerade kommandon. Koden var mest algoritmer för att detektera riktning på ljuskälla samt enkla kommandon på adress- och databuss.

4.2 Avslutande kommentarer

Projekt har enligt vår mening varit väldigt nyttigt och lärorikt. Vi har fått klart ökad insikt i hur ett datorsystem är uppbyggt och implementeras samt alla signaler som är nödvändiga för att sköta trafik över bussar. Vi är klart nöjda med vad kursen gett oss och vill varmt rekomendera den. Slutligen vill vi tacka vår handledare hjälp och stöttning under vägen.

A Kretsschema



B C-kod

```
/*************************************************************************
                          test.c
                          ------
   Detta ar huvudprogrammet som ar skrivet i ANSI C. Exekveringen av hela
   programpaketet borjar i pmain.68k (lage __main).

   exp4() anropas fran assemblyprogrammet exp4.68k vid avbrott.

   _avben() anropar avben.68k vilket tillater avbrott fran PI/T.
*************************************************************************/
unsigned short int *stegm;   /* Ett annat satt att definiera adresser */
unsigned short int *button;
unsigned short int *intack;
unsigned short int *adstart;
unsigned short int *adread;
unsigned short int adflag;
unsigned short int ticcount;
int ft[4];        /*vektor med medelvärden för de 4 fototransistorerna*/
short int countft0;        /*räknare för medelvärdesbildning*/
short int countft1;
short int countft2;
short int countft3;
int ft0[4];        /*vektor med 4 senaste värdena från adomvandlarna*/
int ft1[4];
int ft2[4];
int ft3[4];
unsigned short int s1;
unsigned short int s2;
int ycount;
int tmpy;
int tmpx;
unsigned short int ch;        /* vilken kanal som adomvandlaren ska omvandla från*/
void counterclockw(unsigned int nbr);        /*funktion som roterar riggen moturs*/
void clockw(unsigned int nbr);        /*funktion som roterar riggen medurs*/
void down(unsigned int nbr);        /*tilt på riggen*/
void up(unsigned int nbr);        /*tilt på riggen*/
void wait();        /*fördröjning för att stegmotorerna ska hinna uppdatera*/
void aadstart();        /*startar adomvandling*/
void aadread();        /*läser från adomvandlare*/
void autox();        /*funktion som med hjälp av värden från ad roterar i x-led*/
void autoy();        /*funktion som med hjälp av värden från ad roterar i y-led*/
void run();        /*huvudprogram*/
main(){
        ticcount = 0;
        countft0 = 0;
        countft1 = 0;
        countft2 = 0;
```

```
        countft3 = 0;
        s1 = 0;
        s2 = 0;
        ycount = 0;
        tmpy = 0;
        tmpx = 0;
        ch = 0;
        stegm = (unsigned short int *) 0x0060000; /*adress till stegmotor*/
        button = (unsigned short int *) 0x0080000;/*adress till knappar*/
        adread = (unsigned short int *) 0x0040000;/*adress till AD*/
        _avben();               /*tillagt avbrott */


        aadstart();
        while(s1 == 0){
                up(1);
        }
        run();
        while(1){                       /* evig loop */

        }
}
void run(){        /*oändligt huvudprogram*/
        while(1){
                if((ft[0]+ft[1])>(ft[2]+ft[3])){        /*kollar största värde i y-led*/
                        tmpy = ft[0]+ft[1];
                }
                else{
                        tmpy = ft[2]+ft[3];
                }
                if((ft[0]+ft[3])>(ft[1]+ft[2])){        /*kollar största värde i x-led*/
                        tmpx = ft[0]+ft[3];
                }
                else{
                        tmpx = ft[1]+ft[2];
                }
                if(tmpx > tmpy){
                        autox();
                }
                else{
                        autoy();
                }
        }
}
void autox(){        /*program som styr x-läget*/
        if(((ft[0] + ft[3]) - (ft[1]+ft[2])) > 10){
                if(ycount <= 55){
                        counterclockw(3);
                }
```

```
                else if(ycount > 55){
                        clockw(3);
                }
        }
        else if(((ft[1]+ft[2])- (ft[0]+ft[3])) > 10){
                if(ycount<=55){
                        clockw(3);
                }
                else if(ycount>55){
                        counterclockw(3);
                }
        }
}
void autoy(){        /*program som styr y-läget*/
        if(((ft[0]+ft[1])-(ft[2]+ft[3]))< -10){
                down(2);
        }
        else if(((ft[0]+ft[1])-(ft[2]+ft[3])) > 10){
                up(2);
        }
}
void aadstart(){
    if(ch == 0){
                adstart = (unsigned short int *) 0x0a0000;  /*startar adomv på kanal 0*/
                *adstart = 0xff;
        }
        else if(ch == 1){
                adstart = (unsigned short int *) 0x0a0001;  /*startar adomv på kanal 0*/
                *adstart = 0xff;
        }
        else if(ch == 2){
                adstart = (unsigned short int *) 0x0a0002;  /*startar adomv på kanal 0*/
                *adstart = 0xff;
        }
        else if(ch == 3){
                adstart = (unsigned short int *) 0x0a0003;  /*startar adomv på kanal 0*/
                *adstart = 0xff;
        }
        return;
}
void aadread(){
    if(ch == 0){
                ft0[countft0] =  *adread; /*räknar upp för att få 4 mätvärden på ch0*/
                countft0++;
                ft[0]=(ft0[0]+ft0[1]+ft0[2]+ft0[3])/4;
                if(countft0 >3){
                        countft0 = 0;        /*nollställer räknaren*/
                }
        }
```

```
        else if(ch == 1){
                ft1[countft1] =  *adread; /*räknar upp för att få 4 mätvärden på ch0*/
                countft1++;
                ft[1]=(ft1[0]+ft1[1]+ft1[2]+ft1[3])/4;
                if(countft1 >3){
                        countft1 = 0;        /*nollställer räknaren*/
                }
        }
        else if(ch == 2){
                ft2[countft2] =  *adread;  /*räknar upp för att få 4 mätvärden på ch0*/
                countft2++;
                ft[2]=(ft2[0]+ft2[1]+ft2[2]+ft2[3])/4;
                if(countft2 >3){
                        countft2 = 0;        /*nollställer räknaren*/
                }
        }
        else if(ch == 3){
                ft3[countft3] =  *adread; /*räknar upp för att få 4 mätvärden på ch0*/
                countft3++;
                ft[3]=(ft3[0]+ft3[1]+ft3[2]+ft3[3])/4;
                if(countft3 >3){
                        countft3 = 0;        /*nollställer räknaren*/
                }
        }
}
void counterclockw(unsigned int nbr){ /*flyttar moturs nbr gånger*/
        static int i;
        for (i=0 ; i < nbr;i++){
                *stegm=0x50;
                wait();
                *stegm=0x90;
                wait();
                *stegm=0xA0;
                wait();
                *stegm=0x60;
                wait();
        }
}
void clockw(unsigned int nbr){ /*flyttar medurs nbr gånger*/
        static int i;
        for (i=0 ; i < nbr;i++){
                *stegm=0x60;
                wait();
                *stegm=0xA0;
                wait();
                *stegm=0x90;
                wait();
                *stegm=0x50;
                wait();
```

```
        }
}
void down(unsigned int nbr){        /*flyttar nedåt nbr gånger*/
        if(s2 == 0){
                static int i;
                for (i=0 ; i < nbr;i++){
                        *stegm=0x05;
                        wait();
                        *stegm=0x09;
                        wait();
                        *stegm=0x0A;
                        wait();
                        *stegm=0x06;
                        wait();
                        ycount++;
                }
        }
}
void up(unsigned int nbr){        /*flyttar uppåt nbr gånger*/
        if(s1 == 0){
                static int i;
                for (i=0 ; i < nbr;i++){
                        *stegm=0x06;
                        wait();
                        *stegm=0x0A;
                        wait();
                        *stegm=0x09;
                        wait();
                        *stegm=0x05;
                        wait();
                        ycount--;
                }
        }
}
void wait(){
        int i;
        for(i=0;i<1000;i++){
        }
}
exp2(){               /* avbrottsprogram när adomvandling skett*/
        adflag = 0;
        aadread();
                ch++;
        if(ch > 3){
                ch = 0;
        }
        adflag = 1;
        return;
}
```

```
exp5(){                                  /* avbrottsprogram för knappar*/
        intack = (unsigned short int *) 0x00C0000;
        ticcount++;
        s1 = 0;
        s2 = 0;
        if(*button & 1<<7){
        up(5);
        }
        else if(*button & 1<<6){
        down(1);
        }
        else if(*button & 1<<5){
        clockw(1);
        }
        else if(*button & 1<<4){
        counterclockw(1);
        }
        else if(*button & 1<<1){
        s1 = 1;
        ycount = 0;
        }
        else if(*button & 1<<2){
        s2 = 1;
        }

        if(adflag == 1 && ticcount > 25){
        aadstart();
        ticcount = 0;
        }
        return;
}
```

C Styrlogik

```
device 22V10
CLK 1
A17 2
A18 3
A19 4
AS 5
DS 6
RW 7
GND 12
CSSRAM 14
CSEPROM 15
CSIOMOTOR 16
CSIOSENSOR 17
STARTALE 18
OEAD 19
OE 20
WE 21
CSIACK 22
DTACK 23
VCC 24

start
CSSRAM/=A17*/A18*/A19*/AS;
CSEPROM/=/A17*/A18*/A19*/AS;
CSIOMOTOR=A17*A18*/A19*/AS;
CSIOSENSOR/=/A17*/A18*A19*/AS;
STARTALE=A17*/A18*A19*/DS;
OEAD=/DS*RW*/A17*A18*/A19;
OE/=/DS*RW;
WE/=/RW*/DS;
CSIACK =/A17*A18*A19*/AS;
DTACK/=/CSSRAM+/CSEPROM+CSIOMOTOR+/CSIOSENSOR+STARTALE+OEAD+CSIACK;
end
```

D Avbrottslogik

```
device 22V10
CLK 1
EOC 4
FC0 5
FC1 6
FC2 7
TIC 8
GND 12
VPA 14
IPL02 15
IPL1 16
RRTIC 20
RTIC 21
REOC 23
VCC 24

start
VPA /= FC0 * FC1 * FC2;
IPL02 /= TIC;
IPL1 /= /TIC * EOC;
RTIC = TIC;
RRTIC /=TIC;
REOC = EOC;
end
```