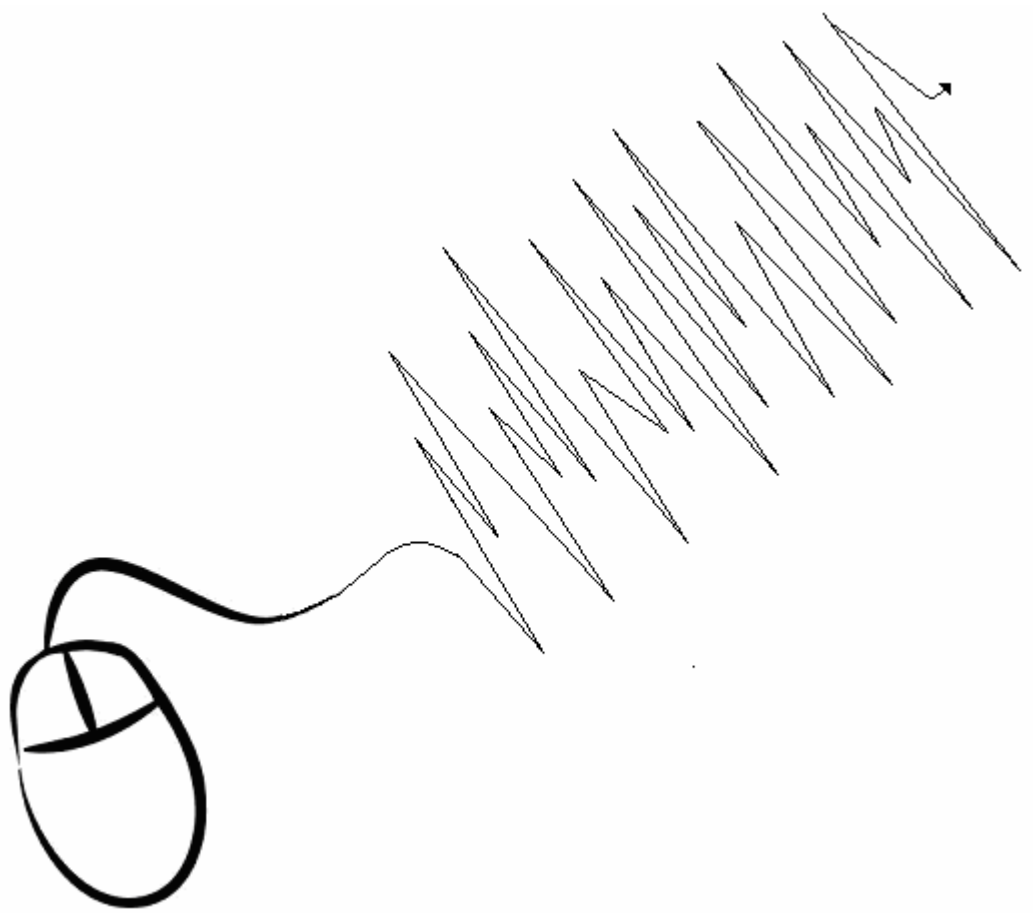
EMG-Styret Mus



Aalborg Universitet
Institut for Elektroniske Systemer
Gruppe E413
30 Maj 2002

Titelblad

Titel: EMG-styret mus
Tema: Mikrodatamatsystemer
Projektperiode: E4 (08.02.2002 til 30.05.2002)

Projektgruppe
E413

Deltagere
Michael Cidlik
Jesper Dueholm
Michael Eriksen
Tommy Jensen
Mads Kelter-Wesenberg
David Langeland
Jacob Larsen

Vejleder
Thomas Graven Nielsen

Censor
Stefan Sörensen

Oplagstal
10

Sidetal
156

Bilag
CD og manual

Afsluttet den
30-05-2002

Synopsis
Denne rapport har til formål, at designe, konstruere og teste et system, der kan emulere en mus ved hjælp af elektriske signaler fra muskler (EMG). Den udviklede PC-software skal fungere i et Windows® operativsystem.

Dette gøres ved at opdele projektet i tre hoveddele, en analog del, en digital del og en softwaredel.
Den analoge del indeholder galvanisk adskillelse, forstærkning, filter samt en konvertering der omdanner de analoge signaler til digitale signaler.
Den digitale del, der består af en mikrodatamat, er opbygget omkring en Motorola 68000 mikroprocessor, der behandler dataene fra den analoge del og omdanner disse til musepakker. Endvidere skal den vha. digitale modstande kunne regulere forstærkningen af EMG-signalet i den analoge del.
Softwaredelen består af et PC-program og et mikrodatamatprogram. PC-programmet er grænse-fladen mellem brugeren og mikrodatamaten. Mikrodatamatprogrammet omdanner EMG-signalerne til musepakker, der kan modtages af en standard musedriver.
Forbindelsen mellem mikrocomputeren og PC'en foregår over en seriel RS232 forbindelse.
Hele systemet er testet og sammenlignet med de krav, der er opstillet i kravspecifikationen.

Konklusionen på rapporten er, at det er lykkedes, at konstruere et system der kan styre en mus. Det er i det pågældende system ikke muligt at kalibrere de enkelte kanaler digitalt.

Summary

The purpose of this report is to design, construct and test a system able of emulating a mouse by means of electrical signals from muscles (EMG). The developed PC-software must be able to run in a Windows® operative system.

This is achieved by separating the system in three parts, an analog part, a digital part and a software part. The analog part contains galvanic separation, amplification, filter and a converter that transforms the analog signals to digital signals.
The digital part, consisting of a microcomputer, is based on a Motorola 68000 microprocessor, which handles the data from the analog part and transforms these to mouse packets. Furthermore it must regulate the amplification of the EMG-signal by means of digital resistors.

The software part consists of a PC-program and a microcomputer program. The PC-program is the interface between the user and the microcomputer. The microcomputer program converts the EMG-signals to mouse packets receivable by a standard mouse driver. A serial RS232 connection is used between the microcomputer and the PC.
The system has been tested and compared to the requirements initially set. The final conclusion on the report states that it was possible to design a system capable of emulating a mouse. It isn't possible to calibrate the channels digitally in the current system.

Forord

Denne rapport er udarbejdet på Aalborg universitet af gruppe E413 i perioden fra den 08.02.2002 til 30.05.2002.

Læsevejledning

Rapporten er opbygget i en hovedrapport samt appendiks i nævnte rækkefølge.
Noter er udarbejdet i fodnoter, der er nummeret fortløbende gennem hele rapporten og henviser til uddybende forklaringer. Kildehenvisningerne er anført i parentes med en forkortelse, der refererer til litteraturlisten, og sidehenvisning til den angivne kilde.
Eks: ….. motoriske endeplade [Fys s. 99-100]....
Kilderne, i litteraturlisten, er opstillet på følgende måde: Forkortelse, Titel, ISBN, Forlag, Udgave, Udgivelse år, Forfatter og evt. Link.
Endelig er nummereringen af figurer og tabeller sket fortløbende gennem de enkelte kapitler i rapporten.
Bilag er vedlagt på medfølgende cd-rom i pdf-format, placeret på bagsiden af rapporten. Til åbning af disse kan Acrobat Reader anvendes. Version 5.05 findes på CD'en. Henvisning til CD'en gøres ved henvisning til den specifikke placering på CD'en.
Eks: ..... [CD\Datablade\AD620.pdf].....

Aalborg Universitet d. 30. Maj. 2002

Michael Cidlik _____

Jesper Dueholm _____

Michael Eriksen _____

Tommy Jensen _____

Mads Kelter-Wesenberg _____

David Langeland _____

Jacob Larsen _____

Indholdsfortegnelse

1 Indledning

De nedenstående 5 punkter beskriver de overordnede formål for projektarbejdet på E-sektorens 4.semester (studieordningen).

1. At give viden om systembegrebet, herunder metoder på anvendelsesniveau til nedbrydning af systemer i delsystemer og forståelse for metoder til beskrivelse af systemer.
2. At give forståelse for metoder til konstruktion af sammensatte digitale systemer, herunder en indføring i grundlæggende digitale kredsløbsblokke, deres brug og begrænsninger.
3. At give forståelse for grundlæggende begreber for datamaters arkitektur.
4. At give viden om programmelsystemer, herunder operativsystemer, udviklingsprogrammel og andet systemprogrammel.
5. At give metoder på anvendelsesniveau til specifikation, konstruktion, realisation, test og dokumentation af programmel og af maskinel.

Til opfyldelse af de ovenstående punkter har gruppen valgt projektforslag nr. 2, med titlen ”EMG-styret mus”.

Det kan ofte være svært for mennesker med visse fysiske handikap, at benytte en standard PC-mus. Formålet med dette projekt er derfor at udvikle en alternativ PC-mus, der gør det muligt for disse mennesker at styre en musemarkør.
En mulighed, der i dette projekt vil blive behandlet, er at anvende EMG-signaler fra brugerens muskler til at foretage denne styring.
I dette projekt, der giver et alternativ til styringen af en musemarkør, vil et komplet system blive designet, konstrueret og testet. Dette alternativ realiseres vha. musklers EMG-signaler, der bruges til at kontrollere musemarkørens bevægelser.

2 Kravspecifikation

2.1 Indledning

2.1.1 Formål

Det kan ofte være svært for mennesker med visse fysiske handikap, at benytte en standard PC-mus. Formålet med dette projekt er derfor at udvikle en alternativ PC-mus, der gør det muligt for disse mennesker at styre en musemarkør.
En mulighed, der i dette projekt vil blive behandlet, er at anvende EMG-signaler fra brugerens muskler, til at foretage denne styring. Princippet er, at EMG-elektroder kan detektere spændinger i en muskel, hvorefter dette signal kan anvendes til at styre musemarkøren på en computerskærm. Produktets endelige navn er således blevet bestemt til "EMG-styret mus".

2.1.2 Referencer

Denne kravspecifikation er udarbejdet efter SPU-modellen[1], som den er beskrevet i "Håndbog i struktureret program udvikling". Endvidere er der vedlagt yderligere information omkring EMG-signalet, museprotokollen og RS-232 standarden i appendiks B.

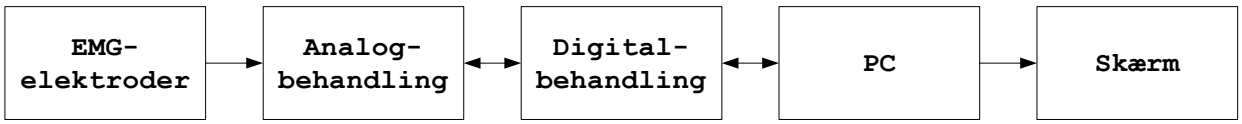
2.1.3 Læsevejledning

Denne kravspecifikation vil indeholde funktionsdiagrammer og blokdiagrammer, der vil blive udspecificeret i mindre funktioner/blokke. Der vil blive opstillet krav til systemet, ud fra behandlingen af de forskellige blokke.

2.2 Generel beskrivelse

2.2.1 Systembeskrivelse

Da systemet modtager et analogt input fra EMG-elektroderne og på baggrund af dette styrer en musemarkør på en skærm, kan systemet inddeles i følgende blokke:



Figur 2.1: Funktionsdiagram over den EMG-styrede mus

EMG-elektroder

EMG-elektroderne skal kunne detektere brugerens muskelspændinger, vha. elektroder, og sende signalet videre. Dette er nærmere beskrevet i appendiks A.

Analog-behandling

Den analoge del indeholder følgende dele:

- *Instrumenteringsforstærkning* der forstærker signalet og fjerner støj.

[1] Struktureret Program Udvikling.

- *Galvanisk adskillelse* der sikrer brugeren mod elektrisk stød.
- *Forstærkning* der forstærker signalet op til et ønsket niveau.
- *Filter* der fjerner støj samt virker som anti-aliasingfilter.

Digital-behandling
Den digitale del indeholder følgende dele:
- *Multiplexer* der multiplexer signalerne til én kanal.
- *A/D-konverter* der konverter et analogt signal til et digitalt signal.
- *Mikrodatamat* der omsætter indgangssignalet til musepakker til PC'en (se afsnit B.1).

PC/Skærm
PC'en modtager musepakkerne fra mikrodatamaten og vha. musedriveren flyttes musemarkøren rundt på skærmen. Endvidere skal PC'en indeholde software til kalibrering af mikrodatamaten.

2.2.2 Systemets funktion
EMG-musen skal for den handikappede være et brugbart alternativ til en standard PC-mus. Princippet vil være, at elektroder placeret på huden af en bruger, vil kunne detektere spændinger i den underliggende muskel. Disse spændinger kan derefter fortolkes til musepakker, efter en behandling af signalet. Dermed skal EMG-musen kunne erstatte musen og anvendes i et Windows® miljø.
Da brugergrænsefladen er 2-dimensionel, og MS Mouse Standard, jf. afsnit B.1, opererer med horisontale og vertikale retningslinier, skal minimum fire muskler benyttes for at genere brugbare signaler. Dette skyldes, at et EMG-signal ikke kan tolkes positivt eller negativt, hvorfor det kræver to muskler at kunne bevæge en musemarkør både frem og tilbage ad en given retningslinie.
Når brugeren således spænder en af de fire muskler, vil musemarkøren bevæge sig i en af de bestemte retninger, op, ned, højre eller venstre.
Alle andre retninger opnås ved at kombinere de fire hovedretninger. Venstre- og højreklik opnås ved at kombinere muskler der styrer modsatrettede retninger.

2.2.3 Systemets begrænsninger
Systemet skal kun understøtte normale højre- venstreklik. Alle andre former for klikfunktioner, samt scroll-funktioner vil ikke blive understøttet af systemet.

2.2.4 Systemets fremtid
I takt med at indsatsen for at få handikappede i arbejde stiger og flere og flere computerbrugere lider af museskader, vil behovet for en alternativ mus stige. For at tilgodese denne udvikling kan systemet i fremtiden gøres mere brugervenligt ved f.eks. at anvende ekstra kanaler i mikrodatamaten, således kliksignalerne kan skilles fra bevægelsessignalerne. Dette kunne gøre det muligt at udstyre systemet med samme funktioner som en normal mus.

2.2.5 Brugerprofil
Systemet henvender sig til handikappede, folk med museskader, og andre der ikke har mulighed for at anvende en standard mus fuldt ud.

2.2.6 Krav til udviklingsforløbet

Kravene til projektets udviklingsforløb er detaljeret beskrevet i E-studienævnets E4 projektenhedsbeskrivelse.
De anvendte programmeringssprog er C og assembler i forbindelse med henholdsvis PC'en og mikrodatamaten.
Afleveringsdato for projektet er torsdag d. 30. maj

2.2.7 Omfang af kundeleverance

Kundeleverancen skal bestå af et fungerende system, der kan erstatte en standard mus på en standard Windows® PC. Software til systemet skal leveres på CD-ROM.

2.2.8 Forudsætninger

Der forudsættes følgende:

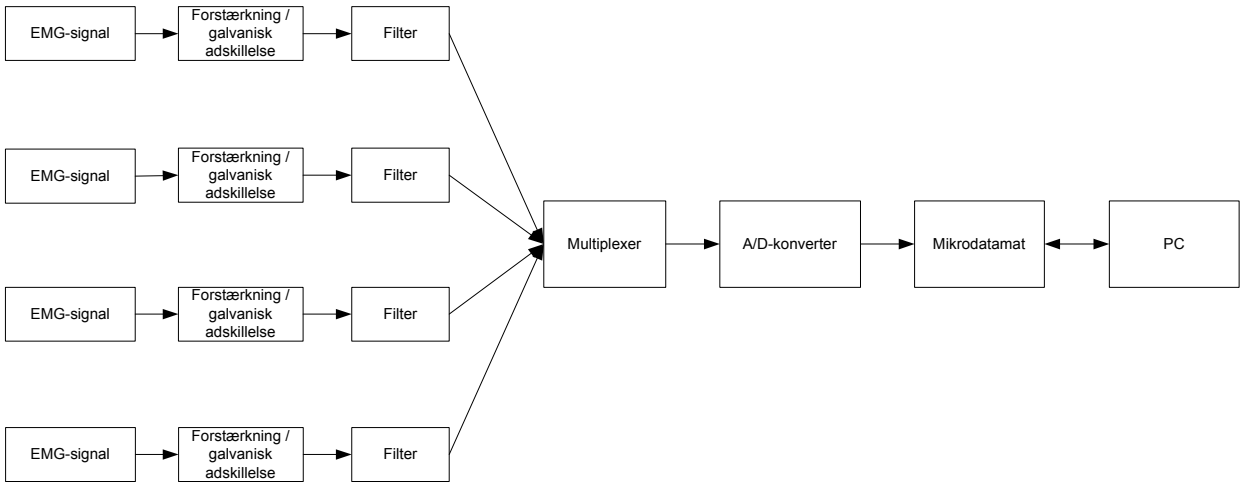
- Mikrodatamaten skal bygges op omkring en Motorola 68 000 mikroprocessor.
- PC'ens serielport skal være RS232-kompatibel, således en MS Mouse Standard kan benyttes.
- PC-softwaren skal skrives til at begå sig i et Windows® miljø.
- Brugeren skal være i besiddelse af og have kontrol over minimum fire fungerende muskler.

2.3 Specifikke krav

Systemet deles ind i software og hardware, der behandles hver for sig.

2.3.1 Hardware

I figur 2.2 ses blokdiagrammet for den EMG-styrede mus. Samtlige blokke behandles enkeltvis.



Figur 2.2: Blokdiagram over den EMG-styrede mus

EMG-signal

EMG-elektroderne detekterer brugerens muskelspændinger og sender signalet videre. Dette signal vil teoretisk ligge på en værdi op til $5\,m\hat{V}$ jf. appendiks A.3. I dette projekt er det dog valgt at benytte $2{,}5\,m\hat{V}$ som det maksimale spændingsniveau. Dette er gjort på baggrund af, at det ved praktiske forsøg har vist sig, at signalet meget sjældent overstiger $2{,}5\,m\hat{V}$.

Forstærkning / galvanisk adskillelse

Da EMG-signalet fra elektroderne er forholdsvis svagt, er det nødvendigt at forstærke signalet op til et brugbart niveau. For at undgå at forstærke støjen op sammen med signalet, er det et krav, at der skal benyttes en forstærkning indeholdende en instrumenteringsforstærker jf. appendiks A.3. Instrumenteringsforstærkerens Common-Mode Rejektion Ratio (CMRR) sættes til 90dB for at opnå et acceptabelt signal/støj forhold.
Da brugeren er i kontakt med forstærkningen igennem EMG-elektroderne, er det nødvendigt at sikre brugeren imod utilsigtet kontakt med lysnettet. Det er derfor valgt at forsyningen, på brugerens side af den galvaniske adskillelse, skal bestå af batterier med et spændingsniveau på ±9V. Forsyningen på den anden side af den galvaniske adskillelse skal bestå af en spændingsforsyning på ± 12V, tilkoblet lysnettet.

Blokkens inputsignal består af EMG-signaler på maksimalt $\pm 2{,}5\, m\hat{V}$, og blokkens outputsignal sættes til maksimalt $\pm 5\hat{V}$. Det er valgt at udstyre blokken med digitale modstande således, at mikrodatamaten kan styre den samlede forstærkning.
Endelig ønskes en lineær forstærkning med en maksimal ulinearitet på $\pm 3dB$.

Filter

Filteret skal fjerne høj- og lavfrekvent støj, og samtidig virke som anti-aliasing filter. Størstedelen af EMG-signalet har en frekvens mellem 20 og 150 Hz, hvorfor det er valgt at benytte et båndpasfilter i dette frekvensområde.

Multiplexer

De fire EMG-kanaler skal multiplexes til én enkelt kanal, således det kun er nødvendig at benytte én A/D-konverter. Eneste krav til multiplexeren er således, at denne skal kunne multiplexe minimum 4 kanaler.

A/D-konverter

A/D-konverterens opgave er at konvertere det analoge EMG-signal til et digitalt signal mikrodatamaten kan bearbejde.
Det er tilstrækkeligt at benytte en 8 bit A/D-konverter, idet museprotokollen (jf. afsnit B.1) opererer med en opløsning på 7 bit. A/D-konverterens samplingsfrekvens fastsættes på baggrund af anti-aliasing filteret.

Endelig skal A/D-konverteringen kunne håndtere signaler på $0\text{-}5\hat{V}$, hvorfor det er nødvendig at implementere et DC-offset og dæmpning af signalet umiddelbar før A/D-konverteringen. Dette gøres idet EMG-signalerne består af såvel positive som negative signaler.

Mikrodatamat

Signalet fra A/D-konverteren skal behandles i mikrodatamaten således, at denne kan omsætte indgangssignalet til musepakker til PC'en. Herunder indgår bl.a. ensretning, midling samt styring af signalforstærkningen umiddelbart efter den galvaniske adskillelse.
Jf. E-studienævnets E4 projektenhedsbeskrivelse skal mikrodatamaten være interruptstyret, og det er endvidere valgt at bygge mikrodatamaten op omkring en Motorola 68k mikroprocessor.

PC

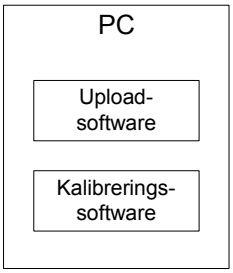
Af hensyn til systemets brugervenlighed er eneste krav til PC'en, at denne skal være en standard PC med serielport.

2.3.2 Software

Systemets software inddeles i to blokke, en mikrodatamat-del og en PC-del. Softwaren til PC-delen udarbejdes i programmeringssproget C, medens softwaren til mikrodatamat-delen udarbejdes på assembler niveau.

PC-software

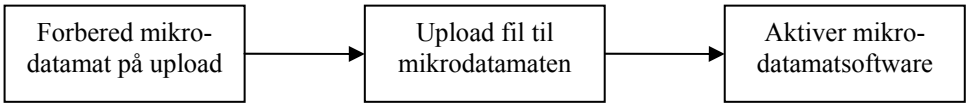
Softwaren til PC'en deles op som illustreret i figur 2.3.



Figur 2.3: Blokdiagram over PC'ens software

Uploadsoftware

Uploadsoftwaren på PC'en har til formål at uploade mikrodatamat-softwaren i mikrodatamatens RAM, og aktivere denne software. Figur 2.4 viser et funktionsdiagram over uploadsoftwarens virkemåde.

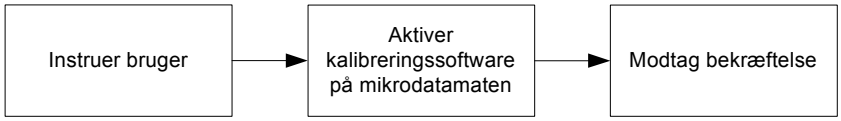


Figur 2.4: Funktionsdiagram over uploadsoftwarens virkemåde

Når uploadsoftwaren aktiveres vil mikrodatamaten forberedes på at modtage en fil. Herefter skal uploadsoftwaren uploade filens data til mikrodatamatens RAM. Når dette er gennemført skal uploadsoftwaren aktivere det software, der nu ligger i mikrodatamatens RAM.

Kalibreringssoftware

Kalibreringssoftwaren på PC'en har til formål at skabe et brugerinterface mellem kalibrerings-softwaren på mikrodatamaten og brugeren. Figur 2.5 viser et funktionsdiagram over kalibreringssoftwarens virkemåde.

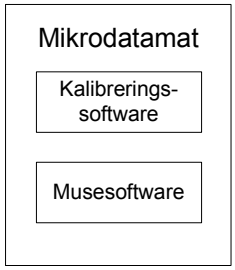


Figur 2.5: Funktionsdiagram over kalibreringssoftwarens virkemåde

Når kalibreringsprogrammet startes op på PC'en, vil brugeren blive præsenteret for en menu, hvor det er muligt at vælge at kalibrere én kanal ad gangen eller bestemme offset og tærskel værdier for samtlige kanaler. Offset værdierne der benyttes i A/D-konverteren, findes for hver kanal, idet disse kan variere og skal benyttes til behandling af EMG-signaler i mikrodatamaten. Tærskelværdierne bestemmes således, at musen ikke reagerer ved støj.

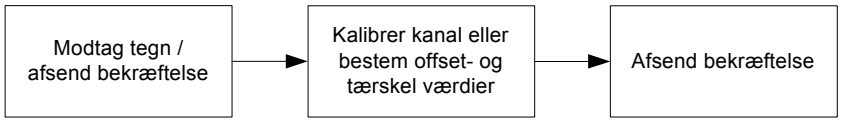
Brugeren vil blive informeret hvorledes denne kalibrering vil foregå og hvad han/hun skal foretage sig. Når brugeren har godkendt instruktionerne afsender kalibreringssoftwaren på PC'en et tegn, der aktiverer og tilkendegiver overfor kalibreringssoftwaren på mikrodatamaten, hvilken kanal der skal kalibreres, eller om offset- og tærskelværdier skal findes.
Kalibreringssoftwaren på PC'en vil afvente en bekræftelse fra mikrodatamaten, når denne påbegynder kalibreringen, eller bestemmelse af offset- og tærskelværdier, og når denne behandling er gennemført.

Mikrodatamatens software
Softwaren til mikrodatamaten deles op i to dele som illustreret i figur 2.6.



Figur 2.6: Blokdiagram over Mikrodatamatens software

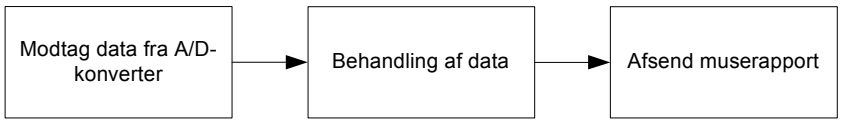
Kalibreringssoftware
Kalibreringssoftwaren har til formål at aflæse amplituden af det modtagne EMG-signal, og ud fra dette bestemme en binær værdi, der benyttes til at indstille forstærkningen, jf. afsnit 2.3.1, via en digital modstand således, at den ønskede forstærkning af signalet opnås. Kalibreringssoftwaren har endvidere til formål at bestemme offset- og tærskelværdier, på baggrund af de modtagne EMG-signaler. Figur 2.7 viser et funktionsdiagram over kalibreringssoftwarens virkemåde.



Figur 2.7: Funktionsdiagram over kalibreringssoftwarens virkemåde

Mikrodatamaten modtager et tegn fra PC'en, der tilkendegiver hvorvidt brugeren ønsker en bestemt kanal kalibreret, eller ønsker offset- og tærskelværdier bestemt. Herefter skal kalibreringssoftwaren afsende en bekræftelse til PC, der indikerer påbegyndt behandling, hvorefter behandlingen påbegyndes. Under behandlingen bestemmes offset- og tærskelværdier eller den aktuelle kanal kalibreres, hvorved den binære værdi i den tilhørende digitale modstand redigeres.
Endelig skal kalibreringssoftwaren afsende en bekræftelse til PC'en, når behandlingen er gennemført.

Musesoftware
Musesoftwarens funktion er at omsætte EMG-signalet fra hver kanal til musepakker. Figur 2.8 viser et funktionsdiagram over musesoftwarens virkemåde.



Figur 2.8: Funktionsdiagram over musesoftwarens virkemåde

Musesoftwaren skal modtage konverterede EMG-signaler fra A/D-konverteren og ud fra disse data generere muserapporter. Muserapporterne skal herefter afsendes til PC'en én musepakke ad gangen. Musepakkerne vil blive genereret i overensstemmelse med MS Mouse Standard som beskrevet i afsnit B.1. Musemarkørens bevægelser og klik er givet ved følgende:

- Kanal 1 vil udgøre en markørændring til højre.
- Kanal 2 vil udgøre en markørændring til venstre.
- Kanal 3 vil udgøre en markørændring op.
- Kanal 4 vil udgøre en markørændring ned.
- Ved samtidig aktivering af kanal 1 og 2 foretages venstreklik.
- Ved samtidig aktivering af kanal 3 og 4 foretages højreklik.

EMG-signalet skal endvidere overstige de bestemte tærskelværdier før den specifikke kanal aktiveres. Dette indføres for at undgå utilsigtede markørbevægelser og klik ved uspændte muskler.

2.4 Eksterne grænsefladekrav

2.4.1 Brugergrænseflade

Brugergrænsefladen for den EMG-styrede mus er udgjort ved brugeren og en PC. Denne brugergrænsefladen skal være udformet på dansk, og musemarkørens retninger i forhold til kanaler skal være udformet som fastlagt i afsnit 2.3.2.

2.4.2 Hardwaregrænseflade

Systemets hardwaregrænseflader består af EMG-elektroderne som grænseflade mellem bruger og hardware, og mikrodatamaten som grænsefladen mellem hardware og PC.
Hardwaren modtager et input i form af EMG-signaler fra muskler, via EMG-elektroderne, og udsender intet output tilbage til brugeren.
Hardwaren benytter en serielport for in- og output ved grænsefladen mellem hardware og PC.

2.4.3 Kommunikationsgrænseflade

Kommunikationsgrænsefladen mellem systemet og PC'en består af en RS-232 standard og en standard Microsoft museprotokol til håndtering af musepakker.

2.4.4 Softwaregrænseflade

Softwaregrænsefladen består af softwaren i mikrodatamaten og softwaren i PC. Softwaren i mikrodatamaten skal være skrevet i assembler, medens softwaren i PC'en skal være skrevet i programmeringssproget C og være baseret til at køre på et Windows® styresystem.

2.5 Krav til systemets ydelse

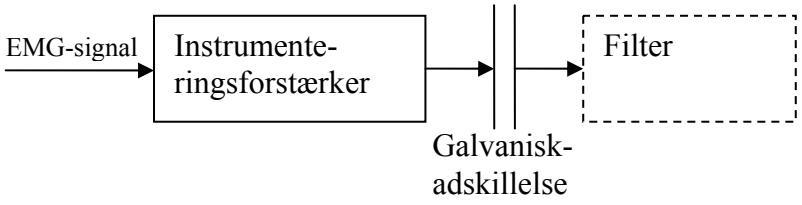
Kravet til systemets respons ved et EMG-signal er, at der ved muskelspændinger skal være en korrekt respons på PC-skærmen. Brugeren må endvidere ikke bemærke en forsinkelse mellem systemets input og output.

2.6 Kvalitetsfaktorer

Da hovedformålet med projektet er at få et produkt til at fungere, lægges der i projektet ikke særlig vægt på at opfylde særlige kvalitetsfaktorer.

3 Forstærkning/galvanisk adskillelse

Til forstærkning af det svage EMG-signal benyttes en instrumenteringsforstærker, der foruden at forstærke signalet også undertrykker støj. Efter denne forstærkning er det nødvendigt at sikre brugeren mod elektriske stød fra det resterende system, hvorfor det er valgt at benytte en galvanisk adskillelse. Der tages derfor udgangspunkt i figur 3.1.



Figur 3.1: Forstærkningen og den galvaniske adskillelses placering i den EMG-styrede mus.

Gennem kravspecifikationen er der blevet opstillet en række krav til forstærkningen med galvanisk adskillelse. Disse krav er opstillet nedenfor:

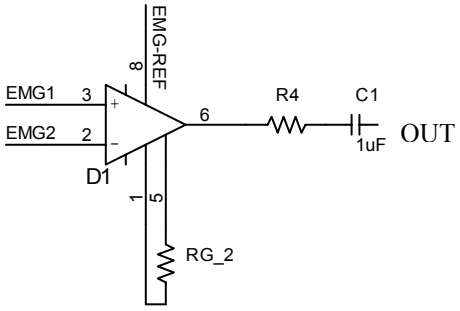
- Lineær forstærkning med en maksimal ulinearitet på $\pm 3dB$
- Minimal Common Mode Rejection Ratio[2] (CMRR) på $90dB$
- Udgangsspænding med en maksimal amplitude på $\pm 5V$

3.1 Instrumenteringsforstærker

Som beskrevet i appendiks A.3 er det nødvendigt at benytte en instrumenteringsforstærker for at minimere støj samt opnå den nødvendige forstærkning.
Instrumenteringsforstærkeren har endvidere flere specifikke krav. Disse er opstillet nedenfor:

- Forstærke EMG-signalet op til en maksimal amplitude på $1,25\hat{V}$, da dette er den typiske værdi den galvaniske adskillelse kan koble.
- Forstærke EMG-signalet lineært op fra $20Hz$ til en frekvens på minimum $150Hz$, idet størstedelen af energien i EMG-signalet ligger i dette frekvensområde.
- Benytte en spændingsforsyning på $\pm 9V$ leveret af batterier.

Det er valgt at benytte instrumenteringsforstærkeren AD620, der på figur 3.2 ses i dens standardopsætning. Denne består udelukkende af forstærkeren selv samt en enkelt modstand.



Figur 3.2: Standardopsætning af AD620

[2] Evnen til at afvise signaler, der er fælles for begge indgange i dB [Zone].

Det er fastlagt, i afsnit 2.3.1, at amplituden af EMG-signalet vil have et maksimalt spændingsniveau på ca. $2{,}5\,m\hat{V}$.
Af hensyn til den galvaniske adskillelse ønskes en maksimal spænding på $1{,}25\hat{V}$ på udgangen af instrumenteringsforstærkeren, hvilket betyder, at forstærkningen i instrumenteringsforstærkeren skal være en faktor 500. Sammenhængen mellem modstanden R_G og forstærkningen i instrumenteringsforstærkeren er givet ved ligning 3.1 [CD\Datablade\AD620.pdf]:

$$R_G = \frac{49{,}4K\Omega}{G-1} \qquad (3.1)$$

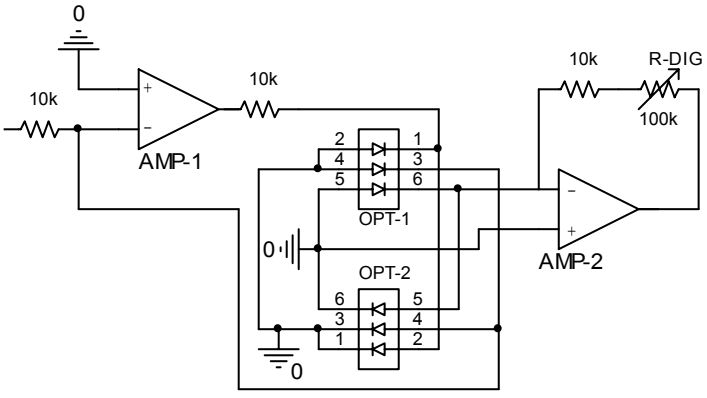
Med udgangspunkt i en ønsket forstærkning på 500 findes værdien af R_G til 98,6Ω. Denne vælges til standardværdien 100Ω.

3.2 Galvanisk adskillelse

Det er nødvendigt med en galvanisk adskillelse mellem brugeren af systemet og lysnettet af hensyn til brugerens sikkerhed.
Kravene til den galvaniske adskillelse er:

- Maksimal ulinearitet på $\pm 3dB$
- Skal kunne håndtere AC-signaler
- Båndbredde på minimum $150Hz$

Valget er faldet på optokobleren IL300 [CD\Datablade\IL300.pdf], da denne lever op til ovenstående krav, samt er tilgængelig i komponentudleveringen. Som udgangspunkt kan den pågældende optokobler ikke anvendes til AC-signaler. Figur 3.3 viser dog en kobling, der gør dette muligt.



Figur 3.3: IL300 I AC-kobling [CD\Datablade\IL300 – Application Notes.pdf]

En anden mulighed havde været at offset-forskyde signalet således, at ingen negative signaler optræder på indgangen af AMP-1. Ulempen ved dette vil dog være, at signalet kun kan forstærkes 250 gange i instrumenteringsforstærkeren af hensyn til den maksimale indgangsspænding på $1{,}25\hat{V}$ i AMP-1. Dette vil resultere i unødigt støj.

Af figuren fremgår det, at koblingen gør brug af to optokoblere samt to operationsforstærkere for hver kanal. Hver af de to optokoblere håndterer henholdsvis positive (OPT-1) og negative (OPT-2) signaler, og de to operationsforstærkere sikrer linearitet og styrer forstærkningen.
I serie med modstanden R2 sættes en $100k\Omega$ digital modstand DS1267 [CD\Datablade\DS1267.pdf], der styres via et serielt signal fra mikrodatamaten, hvilket behandles i afsnit 6.5.4. Dette giver mulighed for en forstærkning på mellem 1,05 og 12,99 jf. ligning 3.2 [CD\Datablade\IL300.pdf]:

$$G = \frac{R2 \cdot K3}{R1} \Rightarrow$$
$$G_{\min} = \frac{10k\Omega \cdot 1,051}{10k\Omega} = 1,05 \quad , \quad G_{\max} = \frac{110k\Omega \cdot 1,181}{10k\Omega} = 12,99 \qquad (3.2)$$

Der tages i ligning 3.2 udgangspunkt i den specifikke optokoblers varierende K3-værdi, jf. databladet. K3-værdien er udtryk for koblingsforstærkningen i optokobleren.

Det er muligt at placere den digitale modstand direkte på instrumenteringsforstærkeren som erstatning for R_G, hvilket vil minimere forstærkning af støj. Dette vil dog betyde, at datasignalerne til den digitale modstand ligeledes skulle indgå i den galvaniske adskillelse, hvilket ville medføre et dyrere og mere kompliceret produkt.

3.3 Delkonklusion og resultater

Forstærkningsgraden i instrumenteringsforstærkeren er testet i appendiks D.1 og fundet til 492, hvilket er fundet tilfredsstillende. Under test viste det sig nødvendigt at indsætte en kondensator i serie med udgangen på instrumenteringsforstærkeren for at undgå DC-offset. Dette resulterer i en dæmpning ved lave frekvenser, hvilket ligeledes er behandlet i appendiks D.1. Det endelige diagram ses i appendiks K.1.
Test af den galvaniske adskillelse i figur 3.3 blev udført med en 100kΩ trimmemodstand i serie med R_2, som erstatning for den digitale modstand, og kondensatoren i serie med indgangen. I appendiks D.1.2 ses testopstilling samt testresultater for beregnede og målte forstærkninger ved forskellige frekvenser.
I tabel 3.1 ses at kravet om linearitet på $\pm 3dB$ er opfyldt, da gennemsnittet af alle målte værdier ligger indenfor dette område. I tabel D.2 til D.5 ses endvidere at samtlige enkeltstående værdier opfylder kravene.

	Kanal 1	Kanal 2	Kanal 3	Kanal 4
20Hz	-2,6dB	-2,6dB	-2,5dB	-2,6dB
100Hz	-0,7dB	-0,6dB	-0,6dB	-0,7dB
150Hz	-0,5dB	-0,6dB	-0,5dB	-0,5dB

Tabel 3.1: Middelforstærkninger for alle kanaler ved henholdsvis 20, 100 og 150Hz

En enkelt worst-case beregning, jf. appendiks D.1.2, levede dog ikke op til kravene, hvilket skyldtes store komponent toleranceværdier. Ud fra resultaterne i appendiks D.1 er forstærkningen dog fundet tilstrækkeligt lineær til at opretholde funktionaliteten i produktet.
Endeligt kan det konkluderes at samtlige krav er opfyldt.

4 Filter

Filteret skal fjerne høj- og lavfrekvent støj og samtidig virke som anti-aliasing filter. Filtrets placering i den EMG-styrede mus ses af blokdiagrammet i figur 4.1

Figur 4.1: Filtrets placering i den EMG-styrede mus

Aliasing skyldes, at frekvenser der er højere end den halve af A/D-konverterens samplingsfrekvens, kan spejles ned og detekteres som frekvenser indenfor området for EMG signaler. Det fremgår af kravspecifikationen, at der benyttes en 8 bit A/D-konverter med en maksimal signalspænding på $5\hat{V}$.
Aliasing forhindres ved, at frekvenser der er højere end den enkelte kanals halve samplingsfrekvens dæmpes mere end det spændingsniveau, der svarer til ½ LSB[3] i A/D-konverteren. Ved en 8 bit A/D-konverter deles spændingsområdet op i 2^8 (256) diskrete niveauer. Da der i det aktuelle tilfælde opereres med en maksimal signalspænding på $5\hat{V}$, bliver forskellen mellem 2 naboniveauer på $19,6m\hat{V}$. Dette ses af ligning 4.1

$$V_{LSB} = \frac{V_{MAX}}{255} = \frac{5\hat{V}}{255} = 19,6m\hat{V} \qquad (4.1)$$

For at undgå aliasing skal støjen, ved den halve samplingsfrekvens, holdes under ½ V_{LSB}. Filtret skal således, jf. ligning 4.2, have en dæmpning på $54dB$.

$$G_{AA} = 20 \cdot \log\left(\frac{½V_{LSB}}{V_{\max}}\right) = 20 \cdot \log\left(\frac{½ \cdot 19,6m\hat{V}}{5\hat{V}}\right) = -54dB \qquad (4.2)$$

På baggrund af kravsspecifikationens krav om en øvre 3dB knækfrekvens ved 150Hz og nødvendigheden af en dæmpning på 54dB ved den halve samplingsfrekvens, bestemmes multiplexerens skiftefrekvens til 1000Hz per kanal. Skiftefrekvensen er således 4000Hz, idet der er 4 kanaler.

Kravene til filtret kan således opsummeres:

- Båndpasfiltret skal have knækfrekvenser ved henholdsvis 20 og 150Hz
- Ved 500Hz skal signalet være dæmpet 54dB

Til realisering af disse krav opdeles filtret i et lavpas- og et højpasfilter.

[3] Least Significant Bit

4.1 Lavpasfilter

Lavpasfiltret skal have en 3dB knækfrekvens ved 150Hz og dæmpe 54dB ved 500Hz. Denne dæmpning er dog kun nødvendig ved worst-case tilfældet, hvor EMG-signalet ved 500Hz opnår en værdi på $V_{\max}$. Figur A.5 viser dog, at denne situation ikke kan forekomme, idet EMG-signalet er betydeligt mindre ved 500Hz. På figur A.5 aflæses en middelværdi for området mellem 20 og 150 Hz til ca. 40 % $V_{\max}$, og middelværdien af EMG-signalet aflæses ved 500 Hz til ca. 6 %. Signalets størrelse ved maksimal indgangssignal på $5\hat{V}$ ved 500 Hz approksimeres således til:

$$\frac{6\%}{40\%} \cdot 5\hat{V} = 0,75\hat{V} \tag{4.3}$$

Signalet er således 16dB mindre ved de 500Hz end i området 20Hz-150Hz, jf. ligning 4.2:

$$20 \cdot \log\left(\frac{0,75\hat{V}}{5\hat{V}}\right) = -16dB \tag{4.4}$$

Signalet skal således, ved 500 Hz, blot dæmpes 54dB-16dB = 38dB. Til beregning af filtrets orden benyttes ligning 4.5:

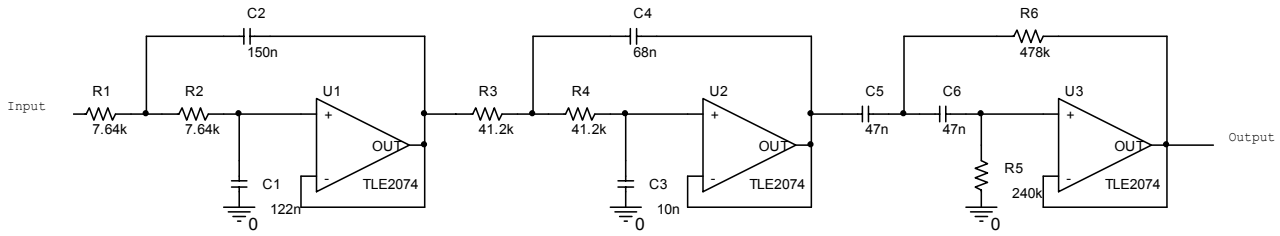
$$dB = n \cdot 20 \cdot \log\left(\frac{\omega_0}{\omega_1}\right) \Leftrightarrow n = \frac{dB}{20 \cdot \log\left(\frac{\omega_0}{\omega_1}\right)} \Rightarrow n = \frac{-38dB}{20 \cdot \log\left(\frac{150Hz}{500Hz}\right)} = 3,63 \tag{4.5}$$

Dette rundes op til et 4. ordensfilter.
Et sådant filter vil ved 500Hz dæmpe 41,8dB, hvilket giver mulighed for et maksimalt indgangssignal på $1,23\hat{V}$ uden at medføre aliasing, jf. ligning 4.2 og 4.5.

Til realisering af lavpasfiltret benyttes to Sallen & Key 2. ordensfiltre, der kobles sammen. Design og udregninger af komponentværdier er foretaget i appendiks E.1.

4.2 Højpasfilter

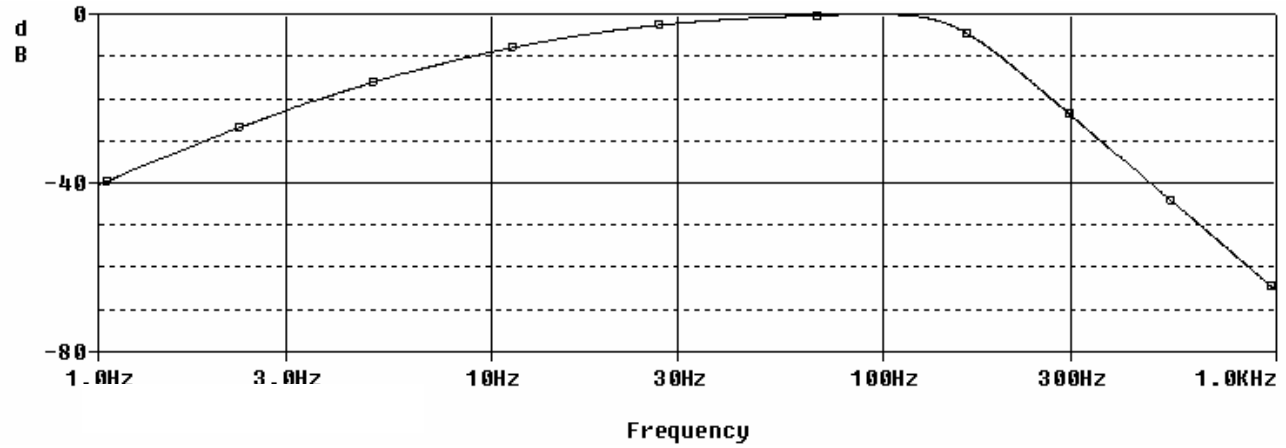
Da det eneste krav til højpasfiltret er en 3dB knækfrekvens ved 20Hz, vælges et Sallen & Key 2.ordensfilter.
Design og udregninger af komponentværdier er foretaget i appendiks E.2.
Diagrammet over det endelige båndpasfilter for en enkelt kanal er vist på figur 4.2.



Figur 4.2: Samlet diagram over båndpasfiltret

4.3 Delkonklusion og resultater

Kravene til båndpasfiltret er 3 dB knækfrekvenser ved henholdsvis 20 og 150Hz, samt en dæmpning på 54 dB ved 500 Hz. Det har dog vist sig at filtret blot skal dæmpe 38dB ved 500 Hz, da signalet allerede er dæmpet 16 dB ved denne frekvens.
Filtret er simuleret i PSpice (se figur 4.3), hvor knækfrekvenserne er aflæst til henholdsvis 24,3Hz og 152,1Hz. Endvidere er dæmpningen ved 500 Hz fundet til 41.6 dB, hvilket opfylder kravet.



Figur 4.3:Pspice simulering af antialiasing filter

På baggrund af test (se appendiks E.3) af filtret kan resultaterne i tabel 4.1 opstilles:

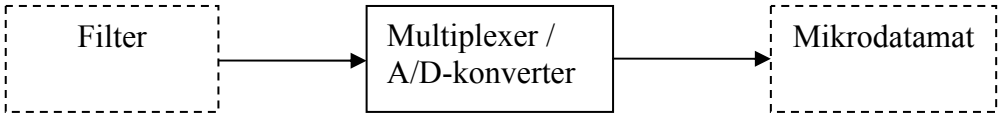
Test 1	Nedre 3 dB frekvens	Øvre 3 dB frekvens	Forventet nedre frekvens	Forventet øvre frekvens	Nedre 3 dB afvigelse	Øvre 3 dB afvigelse
Kanal 1	23,5	155	20	150	13%	3%
Kanal 2	22,8	149	20	150	12%	1%
Kanal 3	23,9	154	20	150	16%	3%
Kanal 4	22,7	152	20	150	12%	1%
Test 2	Signal ved 500 Hz v/ $V_{in} = 1V$	Dæmpning i dB	Forventet dæmpning i dB		Afvigelse	
Kanal 1	0,008	-41,9	-41,8		0,2%	
Kanal 2	0,007	-43,1	-41,8		3%	
Kanal 3	0,008	-41,9	-41,8		0,2%	
Kanal 4	0,008	-41,9	-41,8		0,2%	

Tabel 4.1: : Resultater for test af antialiasing filter

På baggrund af test kan det konkluderes at resultaterne har en minimal afvigelse, hvorfor de er fundet tilfredsstillende. De minimale afvigelser der forekommer kan skyldes unøjagtighed i komponenterne.

5 Multiplexer og A/D-konverter

Multiplexerens opgave er at multiplexe de fire EMG-kanaler til én kanal således, at det kun er nødvendigt at benytte én A/D-konverter. A/D-konverterens opgave er at konvertere det analoge EMG-signal til et digitalt signal, mikrodatamaten kan bearbejde. Multiplexerens og A/D-konverterens placering i den EMG-styrede mus ses af blokdiagrammet i figur 5.1.



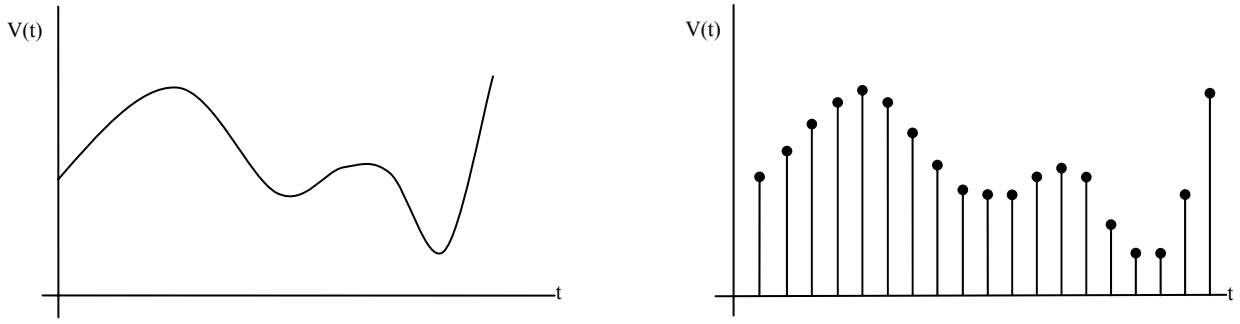
Figur 5.1: Multiplexerens / A/D-konverterens grænseflader.

Da eneste krav til multiplexeren er, at den skal kunne understøtte 4 kanalers multiplexing, er det her valgt først at undersøge begrænsninger og muligheder vedrørende A/D-konverteringen.
Gennem kravspecifikationen og afsnit 4 omhandlende anti-aliasing filtret, er der blevet opstillet en række krav til A/D-konverteren. Disse krav er opstillet nedenfor:

- A/D-konverterens opløsning er valgt til 8 bit
- A/D-konverterens samplingsfrekvens skal være på 4000Hz, 1000Hz pr. kanal
- A/D-konverterens indgangssignal er valgt til ikke at overstige $5\hat{V}$

5.1 A/D-konvertering

A/D-konvertering foregår ved at et analogt indgangssignal konverteres med en bestemt opløselighed, et antal bit, hvorved et tidsdiskret udgangssignal signal opnås. Udganssignalet består således af et givent bitmønster, hvorfor det nu kan benyttes i digitalt øjemed.
Figur 5.2 illustrerer sammenhængen imellem det analoge signal og det resulterende tidsdiskrete signal:



Figur 5.2: Sammenhængen imellem det analoge signal og det resulterende tidsdiskrete signal.

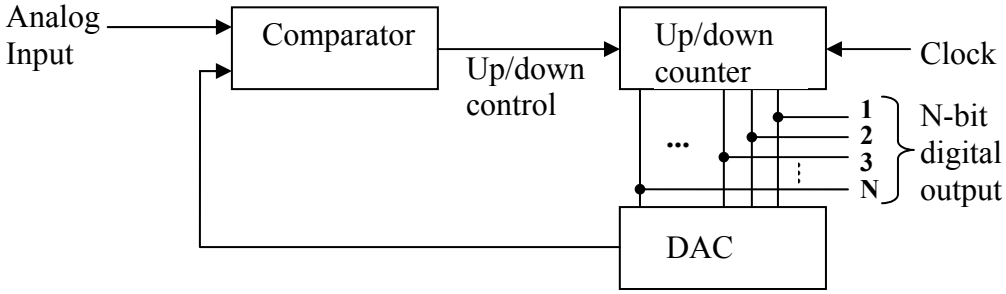
Længden af hver enkelt søjle i det tidsdiskrete signal i figur 5.2 repræsenterer en binærværdi af det analoge signal, til præcis det tidspunkt hvor den givne sampling fandt sted.

5.1.1 A/D-konverteringsmetoder

Der findes adskillelige metoder til at foretage den egentlige A/D-konvertering. To af disse metoder vil blive behandlet her, og er henholdsvis feedback-type konverteren og successiv approksimations konverteren.

Feedback-type konverteren

Feedback-type konverterens virkemåde er illustreret i figur 5.3.



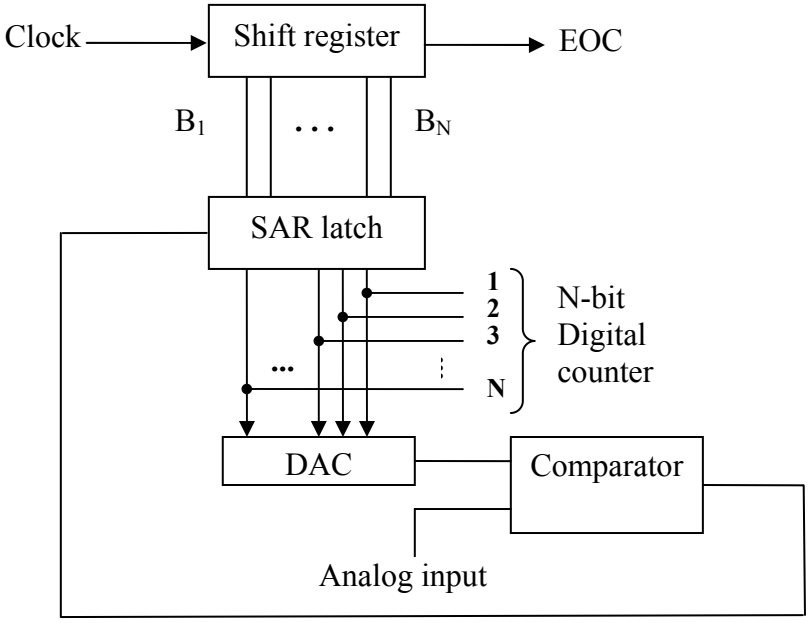
Figur 5.3: Blokdiagram over feedback-type konverteren[Sedra s. 865]

Comparatoren giver som output et kontrolsignal der kan antage den binære værdi 0 eller1, afhængigt af de to analoge signaler. 0 hvis differenssignalet er negativt, og 1 hvis differenssignalet er positivt. Up/down-counteren tæller binært enten ned eller op, afhængigt af om den modtager den binære værdi 0 eller 1 fra comparatoren. Den aktuelle binære værdi sendes til output, og til DAC'en[4]. DAC'en konverterer den binære værdi til et analogt signal, der går til en indgang på comparatoren. Dette analoge signal sammenlignes i comparatoren med det analoge inputsignal. Processen fortsætter indtil DAC'ens output når værdien af det analoge inputsignal. Herefter vil comparatoren og up/down-counteren tælle op og ned omkring det analoge inputsignal, hvorved outputtet vil være tilnærmelsesvis digitalt ækvivalent med det analoge inputsignal.
Ulempen ved at benytte denne type A/D-konvertering er, at konverteren er forholdsvis langsom til at opnå det korrekte digitale niveau, hvis konverterens up/down-counter er på et niveau langt fra det analoge inputsignal. Dette vil ofte være tilfældet, hvis der benyttes multiplexing i forbindelse med A/D-konverteringen, idet niveauerne af de analoge inputsignaler ofte vil være forskellige.
Fordelen er til gengæld, at konverteren er forholdsvis hurtig til at registrere ændringer i inputsignalet, og omsætte dette til det korrekte digitale output, når først det digitale output er ækvivalent med det analoge inputsignal.
På baggrund af ovenstående kan det således konkluderes, at konverteringstiden for en feedback-type A/D-konverter varierer.

Successiv approksimations konverteren

Successiv approksimations konverterens virkemåde er illustreret i figur 5.4.

[4] Digital to Analog Converter



Figur 5.4: Blokdiagram over successiv approksimations konverteren[ADC s. 8]

Successiv approksimations konverteren har til opgave, ud fra et givent analogt inputsignal, at approksimere et digitalt signal med en nøjagtighed svarende til det antal bit konverteren opererer med. Dette gøres ved et foretage en approksimation for hver cyklus, således at approksimationen efter N cykluser har en nøjagtighed af samme antal af bit.
Til at forklare denne virkemåde tages udgangspunkt i et eksempel:
SAR[5] latchen sættes til at operere med 4 bit og en referencespænding på 10V. Det analoge inputsignal sættes til den vilkårlige værdi 8,20V, og eksemplet starter ved begyndelsen af en konvertering.
SAR latchen sender den binære værdi 1 0 0 0 til output såvel som DAC'en, hvor den binære værdi konverteres til det analoge signal 5V, idet referencespændingen er 10V. De 5V sammenlignes i comparatoren med inputsignalet på 8,20V, hvorved comporatoren sender værdien 1 videre til SAR latchen, da 8,20V er større end 5V. På baggrund af dette beholder SAR latchen MSB[6] høj. Det binære tal 1 1 0 0 sendes nu til DAC'en, hvor det konverteres til det analoge signal 7,5V. Når comparatoren nu sammenligner de to analoge signaler 7,5V og 8,20V, vil denne sende værdien 1 videre til SAR latchen, der således beholder anden mest betydende bit høj. Det binære tal 1 1 1 0 sendes nu til DAC'en, hvor det konverteres til det analoge signal 8,75V. Comparatoren sammenligner nu de to analoge signaler 8,75V og 8,20V, og sender derved værdien 0 videre til SAR latchen, der på baggrund af dette sætter næst mindst betydende bit tilbage til 0. SAR latchen sender nu det binære tal 1 1 0 1 til DAC'en, hvor det konverteres til det analoge signal 8,125V. Comparatoren sammenligner nu de to analoge signaler 8,125V og 8,20V. Da 8,20V er større end 8,125V, sendes værdien 1 til SAR latchen, der således bibeholder LSB[7] på 1.
Det digitale outputsignal vil således blive det binære tal 1 1 0 1, svarende til værdien 8,125V $\left(5V + 2,5V + 0,625V\right)$.

Øges antallet af bit i A/D-konverteren vil nøjagtigheden af approksimationen af det digitale outputsignal samtidig øges. Til gengæld øges antallet af nødvendige cykluser tilsvarende.

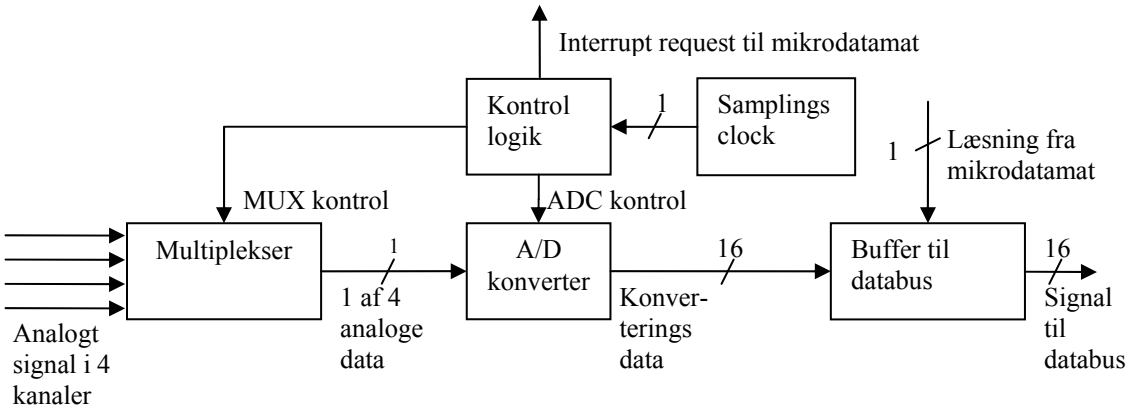
[5] Succesive Aproximation Register
[6] Most Significant Bit
[7] Least Significant Bit

Benyttes outputtet direkte, vil bitmønstret ændre sig fra MSB og nedad i takt med at hver cyklus gennemføres. Outputtet har dog ingen rigtig værdi før samtlige cykluser er gennemført og også LSB er bestemt.
Det direkte output opererer parallelt, og det interne skifteregister giver mulighed for en seriel udlæsning af bitmønstret.
Fordelen ved denne form for A/D-konvertering er, at konverteringshastigheden er konstant.
Dette er samtidig en ulempe, idet denne A/D-konverter aldrig vil kunne opnå samme konverteringshastighed som eksempelvis feedback-type konverteren.

I projektet vælges det at benytte en A/D-konverter, der gør brug af successiv approksimation. Dette gøres som følge af et ønske om en konstant konverteringshastighed. Endvidere vil feedback-type konverteringsmetoden ikke være ideel, idet der i projektet gøres brug af multiplexing.

5.2 A/D-konverteringssystem

A/D-konverteren er afhængig af diverse hjælpekredse. Et blokdiagram for A/D-konverteren med hjælpekredse kan ses på figur 5.5.

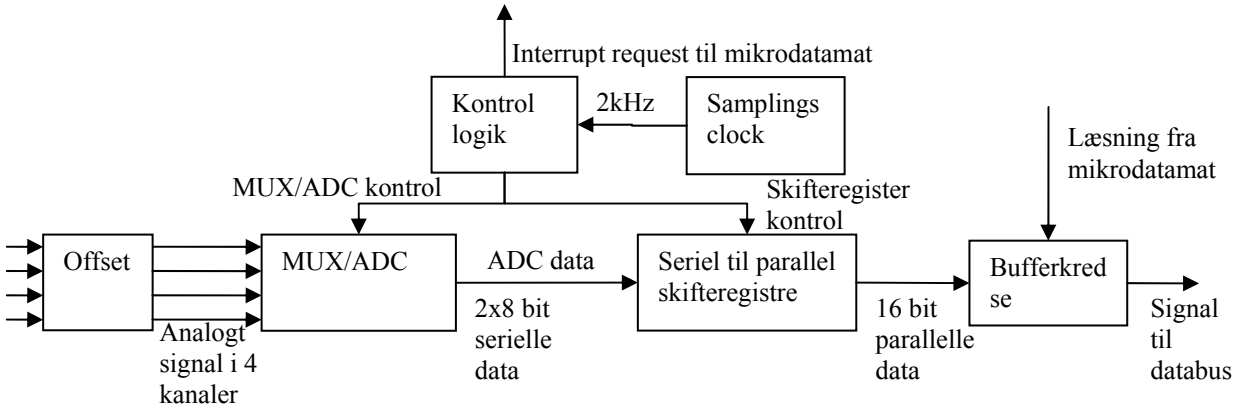


Figur 5.5: Blokdiagram over den overordnede A/D-konverteringskreds.

De 4 analoge signaler sendes ind i multiplekseren, der vælger en af kanalerne på baggrund af en clock, og sender dette signal videre til A/D konverteren. Dataene konverteres, og sendes videre til bufferkredse. Herefter kan de hentes af mikrodatamaten ved at aktivere bufferkredsene.
Kontrollogikken sørger for at sende styresignaler til multiplekser og A/D-konverter.
De fire enheder i figur 5.5 samt clocken vil følgende blive behandlet enkeltvis.

5.2.1 Valg af A/D-konverter

Det valgt at benytte A/D-konverteren ADC838[CD\Datablade\ADC838.pdf], da denne opfylder samtlige krav. Endvidere er der i denne kreds indbygget multiplexer. Denne A/D-konverter har desværre den ulempe, at den konverterede værdi udlæses serielt. Dette problem løses dog ved at benytte et serielt til parallelt skifteregister. Ud over dette vælges det at læse data fra 2 konverteringer ud samtidig. Dette skyldes, at databussen til den anvendte mikrodatamat er på 16 bit. Det betyder også, at kravet til samplingsclocken bliver 2kHz, da hver clockpuls så vil forårsage 2 konverteringer. Da ADC838 ikke kan konvertere negative signaler er det nødvendigt at offset forskyde de fire EMG-signaler, inden disse når A/D-konverteren.
Derved kan blokdiagrammet revideres til det, der ses på figur 5.6



Figur 5.6: Blokdiagram for A/D konverteringskreds med A/D konverter og multiplekser integreret.

Interfacet imellem A/D-konverteringssystemet og mikrodatamaten skal være interrupt styret. Dette opnås ved, at der sendes interrupt til mikrodatamaten når en konverteringscyklus er færdig. Det vælges at sende to forskellige interrupts afhængigt af, om de konverterede data stammer fra kanal 1 og 2 eller kanal 3 og 4 på multiplekseren.

Virkemåde

Det overordnede A/D-konverteringssystem har følgende virkemåde:

1. Samplingsclocken aktiverer A/D-konverteringen. Et internt register definerer hvilke kanaler, der skal samples på. Dette register skifter stilling i takt med samplingsclocken.
2. A/D-konverterens chipselect sættes lav af kontrol logikken, og der læses set-up data (se evt. senere afsnit om digitale input) serielt ind i A/D konverteren, hvorefter konverteringen starter.
3. Det konverterede signal læses serielt ind i de serielle til parallelle skifteregistre. Set-up dataene konfigureres nu i kontrol logikken således, at en tilsvarende måling foretages for den næste kanal.
4. Der vil således ligge 2 gange 8 bit i skifteregistrene, og dermed også i bufferkredsene. Når det er sket, aktiveres et interrupt til mikrodatamaten, der fortæller, at data er klar. Der bruges 2 forskellige interrupts afhængigt af, om der er tale om data fra de høje eller de lave kanaler. Punkt 1-4 vil nu gentages i takt med samplingsclocken.

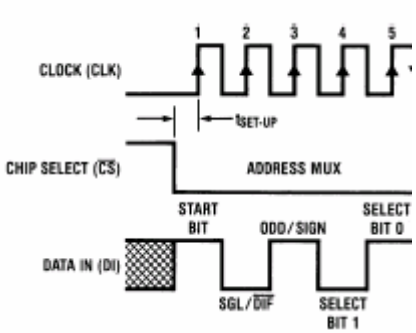
Analoge input

På den analoge side af A/D-konverteren er der følgende input:

- 4 signalindgange, der skal have signal, der ligger mellem V_{CC} og GND. Ubrugte signalindgange kobles til analog stel for at undgå forstyrrelser.
- A/D-konverteren er forsynet med en COM indgang, der bruges som GND ved måling af enkeltsignaler. Da dette er tilfældet her kobles COM til GND.
- V_{REF} er den referencespænding, der konverteres efter. Dette er den spændingsforskel mellem signalindgang og COM, der giver værdien 11111111 ud af A/D konverteren. Den sættes her til 5V, og kobles til analog V_{CC}.
- V^+ benyttes ikke her. Det er en diodebeskyttet indgang til forsyningsspænding og skal ikke kobles til noget. Den er internt koblet til V_{CC}.

Digitale input

A/D-konverteren skal have en række input, der definerer hvordan kredsen skal operere. Disse data sendes serielt ind på DI fra kontrollogikken. Det ses af figur 5.13, hvordan dette signal skal komme i forhold til chipselect og clock.



Figur 5.7: Timing-diagram over clock, chipselect og data in(set-up data).[CD\Datablade\ADC838.pdf]

Inputtene skal komme i følgende rækkefølge:

1. Høj startbit. Definerer starten på konvertering sammen med chipselect til A/D-konverteren. Chipselect styres fra kontrol logikken, og er i A/D-konverteren aktiv lav.
2. Valg af operations mode. Høj bit betyder læsning på 8 enkeltstående kanaler i forhold til fælles stel, mens lav bit betyder læsning på 4 differentielle signaler parvis mellem kanalerne. I dette system, skal denne bit være høj, da der måles på enkeltstående signaler.
3. Valg af lige eller ulige kanaler. Det er valgt at koble signaler ind på de lige kanaler (0, 2, 4 og 6), så denne bit skal være lav i dette tilfælde.
4. Select bit 1 og bit 0. Disse to bit afgør hvilken af fire kanaler A/D-konverteren skal sample. Disse bestemmes af kontrollogikken på baggrund af sampleclocken og kontrol logikkens interne tællere.

Efter indlæsning af disse 5 bit ignoreres alle data på denne indgang, indtil der startes en ny konvertering. Det er umiddelbart efter indlæsning af disse data, at konverteringen starter.
Ud over indgangen DI, findes også indgangene SE[8] og CS. Aktiveres SE ikke, sendes data ud med MSB først. Ønskes derimod data med LSB først kan SE aktiveres. Da data med LSB først ikke skal bruges her, forbindes SE, der er aktiv lav, fast til højt signal.
CS er A/D-konverterens chipselect. Dette er også et aktivt lavt signal, og hvis CS er høj, er alle interne registre nulstillede. Når CS går lav, starter konverteringsproceduren, og kredsen forventer set-up data på DI.

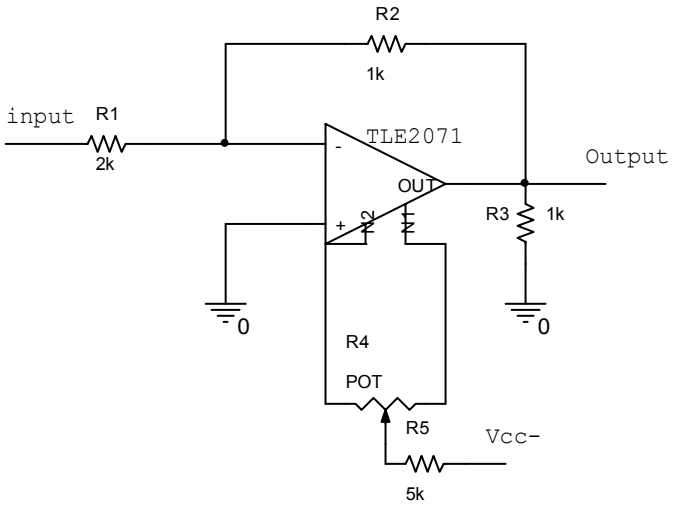
Output

Når set-up data er indlæst i A/D-konverteren, starter konverteringen, og fra næste clockcyklus begynder data at blive læst ud af det serielle output DO. Først kommer et højt startbit, hvorefter data følger med MSB først. Det kan udnyttes, at dette output er i tristate når der læses set-up data ind på DI. Dette betyder, at udgangen bliver højimpedant, og derved vil den ikke påvirke signaler på ledningen, der er koblet til udgangen. Derved kan disse ben kobles sammen, og styres fra samme I/O pin i kontrol logikken. Dette skyldes, at DI som indgang altid er højimpedant.
Kredsen har et status ben, SARS, der er højt så længe konverteringen foregår. Det er i tidsrummet fra startbit'en kommer til LSB er ude. SARS er en udgang og bliver ikke brugt her.

5.2.2 Offset

Da A/D-konverteren ikke kan konvertere negative signaler er det nødvendigt at offset forskyde signalet.
Dette er gjort med operationsforstærkeren TLE2071, der giver mulighed for at indstille offsettet vha. modstanden R4 og potmetret R5[CD\:Datablade\TLE2072.pdf] jf. figur 5.8.

[8] Shift Enable



Figur 5.8: Diagram offset og dæmpning

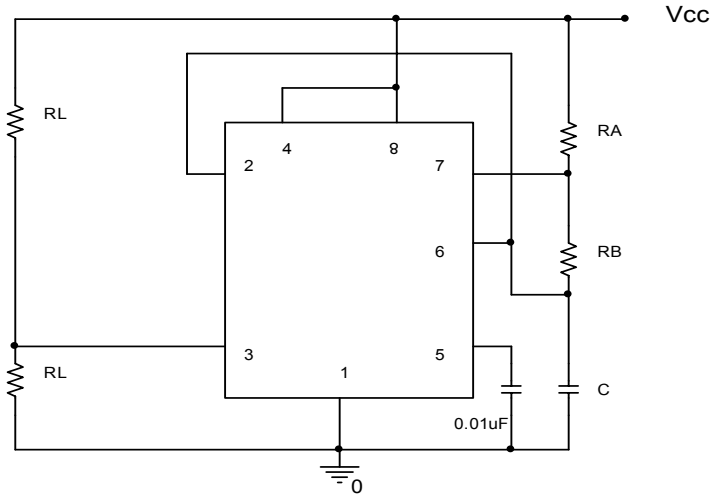
Offsettet sættes til 2,5V da referencespændingen til A/D-konverteren er 5V. Dette betyder at det er nødvendigt at dæmpe EMG-signalet med en faktor 2 således at hele signalet svinger mellem 0 og 5V. Dæmpningen sker vha. modstandene R1 og R2 i en inverterende forstærkning jf. ligning 5.1.

$$G = -\frac{R2}{R1} \Rightarrow G = -\frac{1k\Omega}{2k\Omega} = -0,5 \qquad (5.1)$$

Endelig er det fundet nødvendig at benytte modstanden R3 som "pull-down" modstand idet test har vist at signalet ellers bliver overstyret. Det skyldes, at den anvendte operationsforstærker kobling skal belastes for at kunne operere korrekt.

5.2.3 Samplingsclock

For at kunne generere en $2kHz$ clock, der kan aktivere sampling i A/D-konverteren, benyttes en 555 timerkreds, der kobles som astabil multivibrator efter anvendelses eksempler i databladet[CD\Datablade\LM555.pdf]. Denne kobling kan ses på figur 5.9.



Figur 5.9: LM555 koblet som astabil multivibrator

Duty cycle har ikke betydning for virkemåden af A/D konverteren. Duty cycle begrænses til gengæld af, at den simple multivibrator, der bruges i dette tilfælde, ikke kan have duty cycle på

50% eller derover, hvilket kan ses af ligning 5.2. Duty cycle vælges til 45%, da det må formodes, at kredsen opererer bedst ved duty cycles omkring 50%.

$$D = \frac{R_b}{R_a + 2R_b} \tag{5.2}$$

Clockens frekvens bestemmes af ligning 5.3, der kan omskrives til ligning 5.4

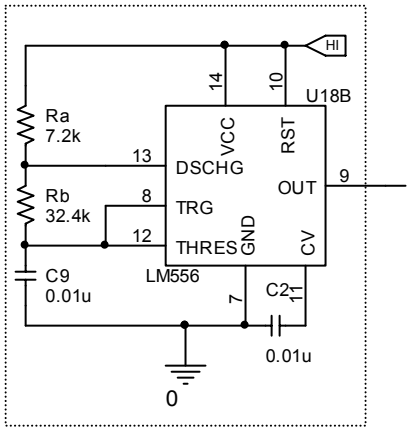
$$f_{clk} = \frac{1.44}{(R_a + 2R_b)C_9} \tag{5.3}$$

$$R_a + 2R_b = \frac{1,44}{f_{clk}C_9} \tag{5.4}$$

Ved at sammenholde ligningerne 5.2 og 5.4, kan modstandene beregnes som i ligning 5.5, når den aktive kondensator C_9 vælges til $10nF$.

$$\begin{aligned} R_b &= \frac{1,44D}{f \cdot C_9} \\ R_b &= \frac{1,44 \cdot 0,45}{2k \cdot 10n} = 32,4k\Omega \\ R_a &= \frac{R_b}{D} - 2R_b \\ R_a &= \frac{32,4k}{0,45} - 2 \cdot 32,4k = 7,2k\Omega \end{aligned} \tag{5.5}$$

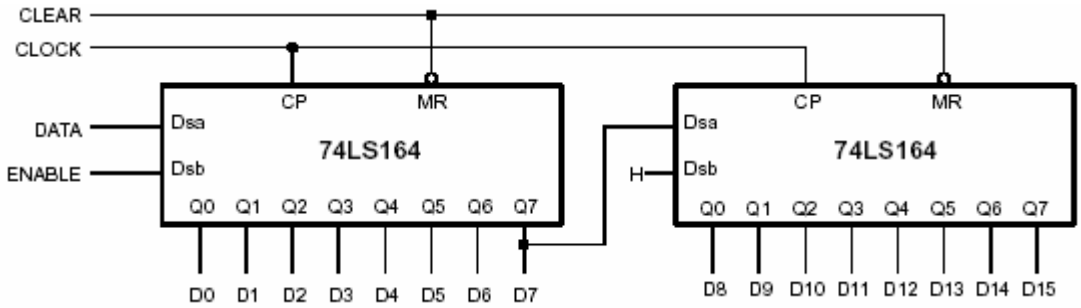
Herefter kommer samplingsclocken til at se ud som på figur 5.10. Den angivne kondensator C_2 på $10nF$ er en støjafkoblingskondensator.



Figur 5.10: Diagram over samplingsclocken med komponentværdier.

5.2.4 Skifteregistre

Da de serielle data skal læses ud på en parallel databus, læses de ind i et sæt skifteregistre, der konverterer fra serielle data til parallelle data. Til dette formål vælges 74LS164, der er beregnet til dette. Da denne kreds er en 8 bit kreds, og der er behov for 16 bit til de 2 kanaler, kobles 2 kredse sammen. Dette er illustreret i figur 5.11.

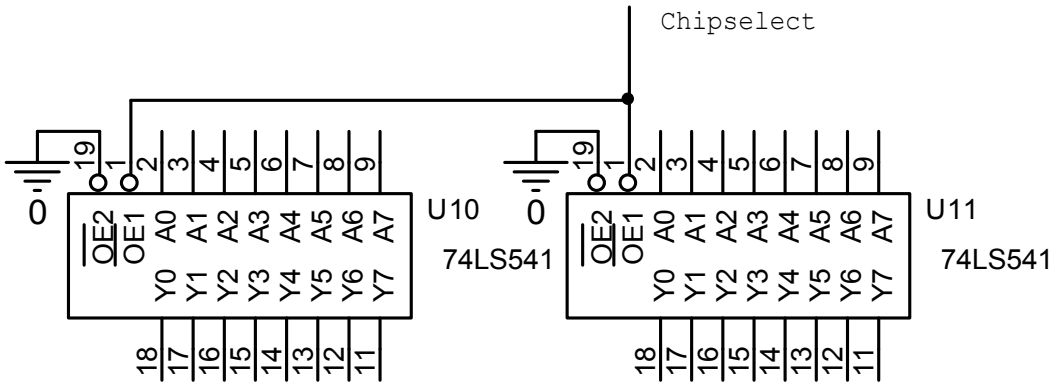


Figur 5.11: Sammenkobling af to 8 bit skifteregistre.[CD\Datablade\74LS164.pdf]

Når de konverterede data skal de læses ud i skifteregistret aktiverer kontrollogikken clocken til skifteregistret samtidig med at der læses data ud. Data læses ud ad A/D konverteren på nedadgående clockpuls, mens de læses ind i skifteregistrene på opadgående. Herved gives en halv clockpuls til at sætte dataene op på udgangen. Dette er tilstrækkeligt, da A/D konverteren bruger 650ns til at sætte dataene op, og den halve clockperiode svarer til 1250ns.
MR er en aktiv lav indgang på skifteregistrene, der benyttes til at resette dataene på skifteregistret. Da dette ikke ønskes forbindes den fast til højt signal. Dsb indgangen på skifteregistret bliver internt AND'et med data indgangen dsa, og kan derved bruges til at enable dataindgangen. Dette bruges ikke her, da den i forvejen enables ved at aktivere og deaktivere clocken. Derfor forbindes indgangen, der er aktiv høj, til højt signal, så den altid enables.

5.2.5 Bufferkredse

For at kunne koble fra databussen, når mikrodatamaten ikke læser på A/D-konverteren, indsættes en bufferkreds. Denne bufferkreds har den egenskab, at dens udgange kan gå i tristate, så den kun påvirker databussen, som den er koblet direkte til, når der er aktiveret et chipselect.
Når bufferkredsen modtager CS lav fra adressedekoderen i mikrodatamaten, kobles høj impedans mode fra og dataene sendes ud på bussen.
Til formålet er valgt kredsen 74LS541[CD\Datablade\74LS541.pdf], der er en standard 8 bit kreds. Da mikrodatamatens databus er en 16 bit parallel bus, og der er 16 bit data fra 2 konverteringer, benyttes to 8 bit kredse, der håndterer en halvdel af bussen hver. Dette opnås ved at koble chipselect for de 2 kredse sammen.
Dette er illustreret i figur 5.12.



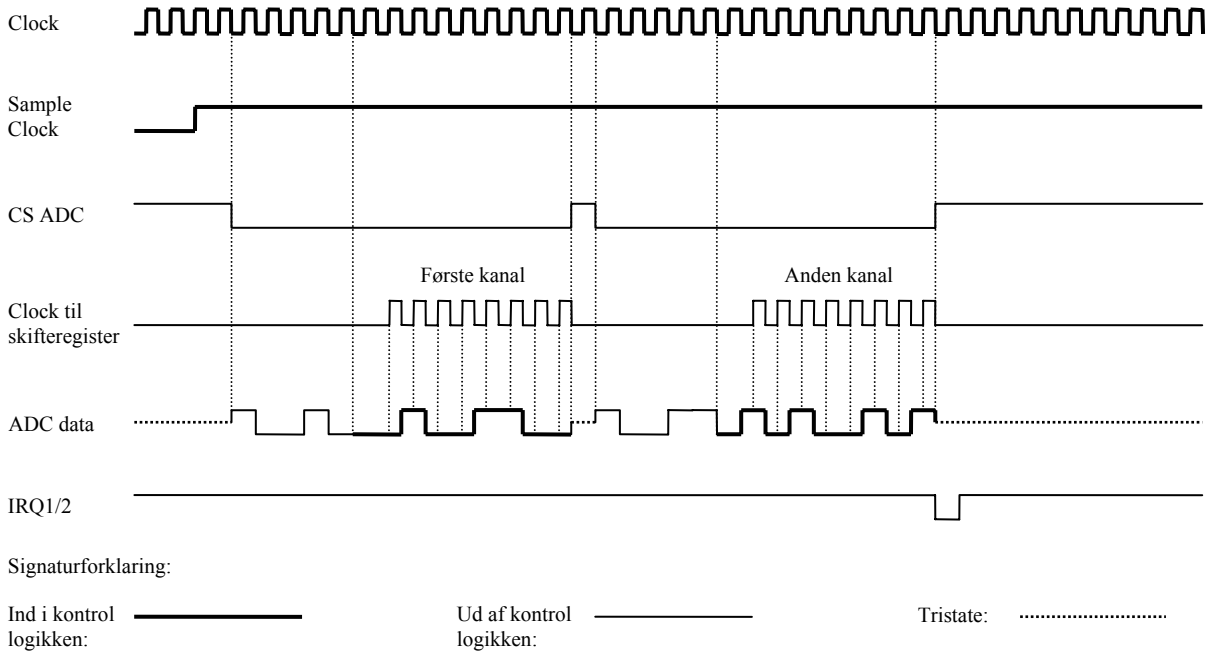
Figur 5.12: Sammenkobling af to 8 bit buffer kredse

5.2.6 Clock

A/D-konverteren kan maksimalt klare en clock på 400kHz [CD\Datablade\ADC838.pdf]. Fra mikrodatamaten er en 800kHz clock tilgængelig [CD\:Datablade\M68000PRM.pdf]. Deles denne med en 2 deler(1 bit tæller), kan mikrodatamatens clock bruges som clocksignal til A/D-konverteren.
2 deleren implementeres i kontrol logikken.

5.2.7 Kontrol logik

På baggrund af den tidligere beskrevne virkemåde, kan timing diagrammet over kontrol logikkens ønskede funktion opstilles, hvilket er illustreret i figur 5.13.



Figur 5.13: Timing diagram over kontrol logikkens output.

For at kunne holde styr på alt dette, er det valgt at placere al logik i en enkelt kreds af typen PA7540[CD\Datablede\PA7540.pdf], der er et såkaldt PEEL Array.
Kredsen er udstyret med 40 registre, hvoraf de 20 er interne. Hvert register kan konfigureres til enten D flipflop, J/K flipflop eller T flipflop. Samtidig kan registrene, modsat en standard 22CV10 kontrolleres af logiske udtryk i kredsen, og ikke kun clocken ind ad clock benet på kredsen.

5.2.8 Programmering af kontrol logikken

For at kunne programmere kontrol logikken er det nødvendigt at kende dennes præcise funktioner samt input og output. Inden behandling af disse foretages opsummeres kort kontrol logikkens ønskede egenskaber:

- Generere en 400kHz clock på baggrund af mikrodatamatens 800kHz clock.
- Starte konvertering i takt med samplingsclocken.
- Styre chipselect til A/D-konverteren.
- Generere set-up data til A/D-konverteren.
- Styre clock til seriel til parallel skifteregistre.

- Sende interruptsignal til mikrodatamat ved afslutning af konvertering.

Ovenstående krav til kontrol logikken realiseres vha. følgende input og output.

Input

Kredsen skal have følgende input:

- Samplingsclock på 2 kHz. Benævnes CLK2K.
- 800kHz clocksignal, der kommer fra en fast udgang på mikrodatamaten. Påtrykkes CLK2 indgangen på PA7540.

Output

Kredsen skal have følgende output:

- Chipselect til A/D-konverter. Benet benævnes CS_ADC.
- Clocksignal til A/D-konverter på 400kHz. Går til CLK på ADC838, og CLK1 på PA7540. Det sidste skyldes, at dette er nær det eneste, der ikke kan udføres med interne forbindelser. Benævnes CLK400K
- Clocksignal til skifteregistre. Går til CLK på skifteregistrene. På PA7540 benævnes denne SR_CLK.
- Set-up data til A/D konverter. Benævnes ADC_DATA
- 2x interrupt request til mikrodatamat. Benævnes IRQ1 og IRQ2.

Funktioner

For at kunne holde styr på konverteringen, og udføre det, der er beskrevet i afsnit 5.2.7, skal følgende funktioner være placeret i PA7540:

- En timer, der holder overordnet kontrol med konverteringen. Der skal bruges 5 bit, så den kan holde rede på i alt 32 states. Registrene benævnes TMR0-4, og funktionen benævnes TIMER. De 32 states benyttes således, hvad der også kan læses af statediagrammet i appendiks F.2:
  - State 0 er neutralt state, hvor der ikke sker noget.
  - 5 states, S1-S5, går til adressering af multiplekseren.
  - 9 states, S6-S14, går til udlæsning af data og startbit fra A/D konverter.
  - 1 states, S15, afsættes til pause i forbindelse med start af en konvertering.
  - 1 state, S16, afsættes til afsendelse af interrupt request.
  - Resten, S17-S31 benyttes ikke her. Det skal kodes således, at hvis et af disse states ved en fejl skulle blive aktivt, er næste state S0, hvor alting resettes.
- En clockdeler, der deler $800kHz$ ned til $400kHz$. Benævnes CLKDIV.
- En JK-flipflop, der holder styr på, hvornår samplingsclocken går fra lav til høj, og konverteringen skal starte. Benævnes TRIG.
- I/O konverteringsdata der består af følgende:
  - 2 D-flipflops, der kan holde rede på hvilken multiplekser kanal, der skal konverteres på næste gang. Benævnes CH0 og CH1.

I appendiks F.2 og F.1 findes henholdsvis state diagram og state tabel over kontrol logikkens virkemåde.

5.2.9 Programmering af PEEL Array

Programmeringen af kredsen PA7540 er foregået i et program, der hedder WinPLACE fra ICT[9]. Dette program indeholder funktioner til at få overblik over den programmerede kreds, og til simulering af det færdige resultat. Logiske udtryk bliver skrevet ifølge dette programs syntaks.

[9] Producenten af kredsen.

Den færdige kildekode er i appendiks F.3, her en gennemgang af funktionerne:

Timer funktion

Til denne funktion er det valgt at bruge programmets indbyggede funktion til at lave state diagrammer, da det giver mere overblik. Et state diagram for denne funktion samt en beskrivelse af hvert state ses i appendiks F.2. Selve funktionen er defineret i koden imellem betegnelsen:

```
STATE_DIAGRAM TIMER
```

og efterfølgende:

```
END
```

samt:

```
DEFINE
TIMER = [T4 T3 T2 T1 T0]
```

Sidstnævnte definerer hvilke registre, der anvendes til funktionen.
Timeren holdes på 0 indtil CS_BUF går fra lav til høj. Når dette sker, startes den cyklus, der er beskrevet i statediagrammet. Herefter resettes, hvorved tælleren holder state 0 indtil næste gang CS_BUF går fra lav til høj.
Detektionen af status for CS_BUF foregår ved at JK flip flop'en "TRIG" sættes når CS_BUF går lav, og resettes ved state 13 under sidste konvertering.

Clockdeler

Clockdeleren er en 1 bit tæller, der er lavet ved at føre den inverterede udgang på D flip flop'en CLKDIV tilbage til indgangen, mens den clockes fra CLK2, der er på 800kHz. Herefter kan udgangen føres ud af benet CLK400K.

Chipselect for ADC838

A/D konverterens chipselect skal være høj, hvor kredsen ADC838 ikke skal arbejde. Det vil sige i state 0, 15 og 16.

I/O konverteringsdata

Som tidligere beskrevet, er det nødvendigt at sætte ADC838 korrekt op, for at den kan fungere. Denne adressering foregår ved timer state S1-S5. Udgangen ADC_DATA er i dette tilfælde defineret således:

1. Startbit. Her er udgangssignalet højt i alle tilfælde.
2. Valg af læsning på differentielt signal kontra enkeltsignaler. Her vælges altid enkeltsignaler, hvilket betyder, at udgangssignalet er højt.
3. Valg af lige/ulige kanaler. Skal altid være lav.
4. Høj bit for valg af multiplekser kanal. Udgangssignalet skal følge status for CH1.
5. Lav bit for valg af multiplekser kanal. Udgangssignalet skal følge registret CH0. Registret CH0 bruges til at definere, om det er første eller anden konvertering i cyklussen, der gennemgåes. Værdien inverteres hver gang state 13 nås. Samtidig resettes registret ved state 0.

Lige efter disse set-up data følger konverteringsdataene fra A/D-konverteren til skifteregistrene. Der er tale om 9 bit, og de tilsvarende 9 states, S6-S14 er defineret ved et startbit, der udlæses ved S6, og 8 databit, der udlæses med MSB først.
I states S0, S15 og S16 skal der holdes pause fra konverteringen, og ADC_DATA skal være i tristate.

Clocksignal for skifteregistre

De serielle til parallelle skifteregistre skal clocke på midten af hver databit, der læses ud af kontrol logikken. Dette kan rimeligt nemt opnåes, da skifteregistrene er forkanttrigget, A/D-konverteren er bagkanttrigget jf. figur 5.13. Skifteregistrene clocker således på midten af hver databit ved, at

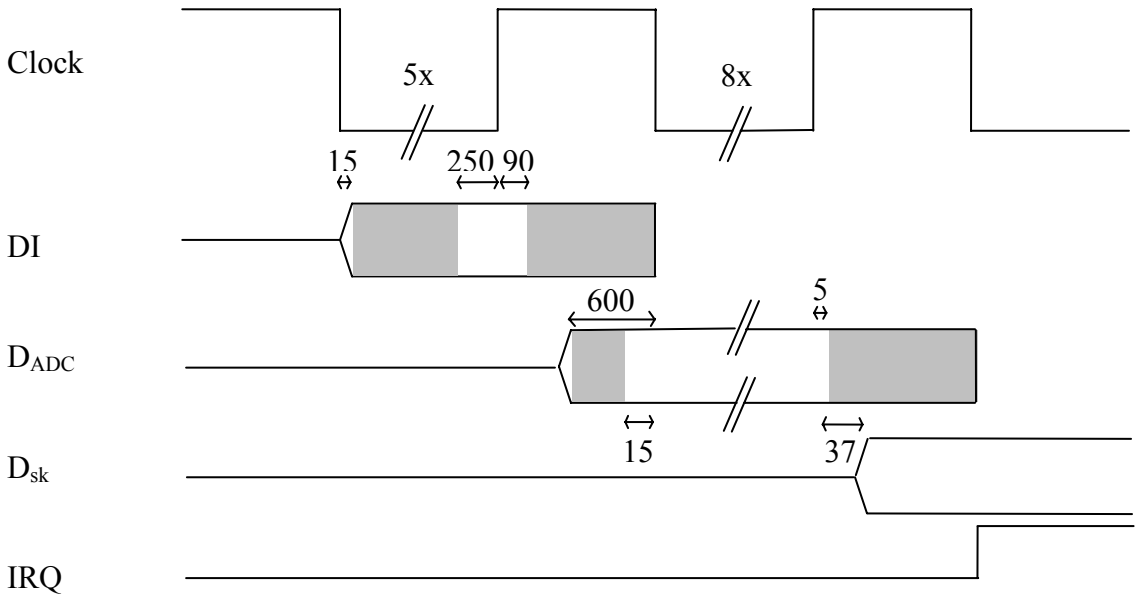
AND'e udtrykket for hvornår der er data på udgangen (state S7-S14) med CLK1, der er den overordnede clock for kredsen.

Afsendelse af interrupt

IRQ1 skal være lav i state S16, når CH1 er lav, hvilket betyder konvertering på kanal 1 og 2 er færdig. IRQ2 skal være lav i state S16 når CH1 er høj, hvilket betyder konvertering på kanal 3 og 4 er færdig. I alle andre states skal de være høje.

5.2.10 Timing

For at systemet kan fungere som beskrevet er det vigtigt, at signalerne er på de respektive indgange når de forventes. Disse ses af figur 5.14:



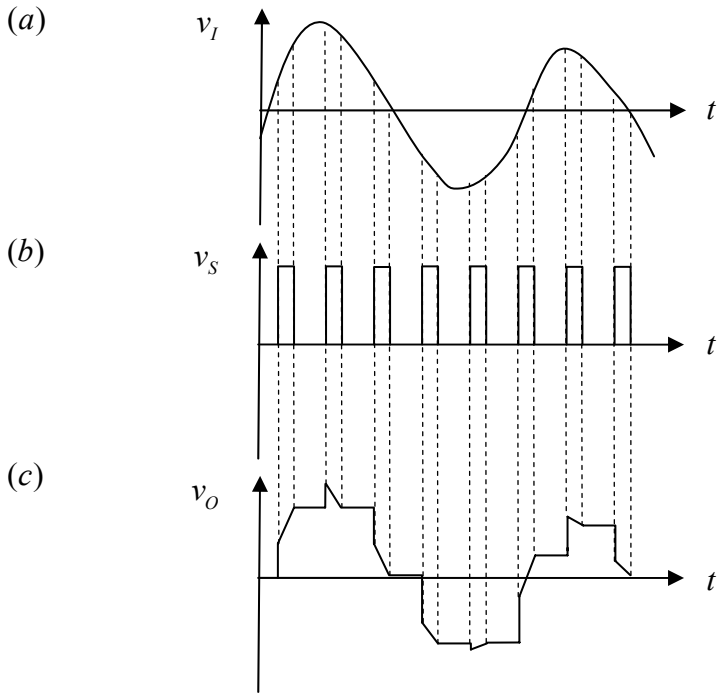
Figur 5.14: Timing i A/D-konverteringen. Tider er i nanosekunder.

- Set-up data skal stå på DI mindst 250ns før clocken går fra lav til høj. De står der 15ns efter clocken går fra høj til lav, hvilket betyder, at de har stået i 1235ns når clocken går høj igen.
- Efter clocken er gået høj, skal data holdes i mindst 90ns. De holdes indtil clocken går lav igen, hvilket betyder, at de holdes i 1250ns.
- Der går 650ns fra clocken går lav til data er klar på dataudgangen af A/D-konverteren. Data skal være klar mindst 15ns før clocken går høj. Her går der 600ns.
- Efter clocken går høj, skal data holdes i mindst 5ns. Her holdes det indtil clocken går lav igen, hvilket betyder, at de holdes i 1250ns.
- Ved sidste databit går der 1250ns fra clocken går høj til der aktiveres interrupt. Data er klar på udgangen af skifteregistrene, dvs indgangen af bufferkredsene efter maksimalt 37ns.
- Når der kommer chipselect fra adressedekoderen skal data være klar på udgangen af bufferkredsene efter maksimalt 227,5ns, jvf. appendiks G.2.1 CPU timing. Bufferkredsene har data klar efter 38ns.

Kontrollogikken giver maksimalt en forsinkelse på 15ns, hvilket ikke kan forskyde på de margener, der ses her.

5.2.11 Sample/hold kredsløb

I forbindelse mede A/D-konvertering skal det afgøres, hvorvidt det er nødvendigt med et s/h kredsløb på A/D-konverterens analoge indgang.
Formålet med et s/h kredsløb er at holde det analoge inputsignal konstant, medens A/D-konverteren foretager en sampling, hvilket kan være en nødvendighed for at opnå en nøjagtig konvertering jf. afsnit5.1.1. Nødvendigheden af et s/h kredsløb afhænger af, hvor meget inputsignalet ændrer sig i løbet af en sampling.
S/h kredsløbets virkemåde er beskrevet i figur 5.15.



Figur 5.15: (a) analog inputsignal. (b) Samplingssignal. (c)Outputsignal til A/D-konverter.

Af figur 5.15 fremgår det, at det analoge inputsignal holdes konstant, i perioden mellem hver samplingssignal. A/D-konverterens samplinger finder sted hver gang signalet holdes konstant, hvorved A/D-konverteren opnår størst nøjagtighed.
Holdes inputsignalet ikke konstant, under én sampling, risikeres det, at inputsignalet ændres til et andet af A/D-konverterens 255 niveauer. Inputsignalet må således, i følge en tommelfingerregel, i løbet af en sampling, ikke ændre sig mere end det der svarer til ½LSB.
Nødvendigheden af et s/h kredsløb afgøres således af inputsignalets frekvens og amplitude, samt A/D-konverterens samplingsfrekvens og opløselighed.
I det følgende beregnes, hvorvidt et s/h kredsløb er nødvendigt i det aktuelle tilfælde. Der tages udgangspunkt i et sinusformet inputsignal, idet alle signaler kan udtrykkes som en sum af sinussignaler:

- Først findes den maksimale hældning for et sinussignal med amplitude på $5\hat{V}$ og en frekvens på 150Hz. Dette signal vælges, som worst-case tilfælde, idet forholdet mellem spændingsniveau og tid er størst ved netop denne frekvens, pga. anti-aliasing filtret.

- Den maksimale hældning for et sinussignal findes ved hver halve periode, når sinussignalets amplitude er 0, hvilket i det aktuelle tilfælde er $\frac{1}{300}s$. Den maksimale hældning er således fundet ved ligning 5.6:

$$V'(t) = \frac{dV}{dt} 5V \sin(2\pi 150 \cdot t) = \frac{25\pi^2 \cdot \cos(300\pi \cdot t)}{3}$$
$$V'\left(\frac{1}{300}s\right) = \frac{25\pi^2 \cdot \cos\left(300\pi \cdot \frac{1}{300}s\right)}{3} = 82{,}1\frac{V}{s} \qquad (5.6)$$

- Det tager 8 clockcyklusser at foretage en konvertering. Da A/D-konverteren benytter en clock på 400kHz, vil en enkelt sampling vare $8 \cdot \frac{1}{400000Hz} = 20\mu s$.
- Herefter bestemmes den maksimale ændring i inputspændingsniveau, der kan finde sted under én sampling:

$$82{,}1\frac{V}{s} \cdot 20\mu s = 1{,}64mV \qquad (5.7)$$

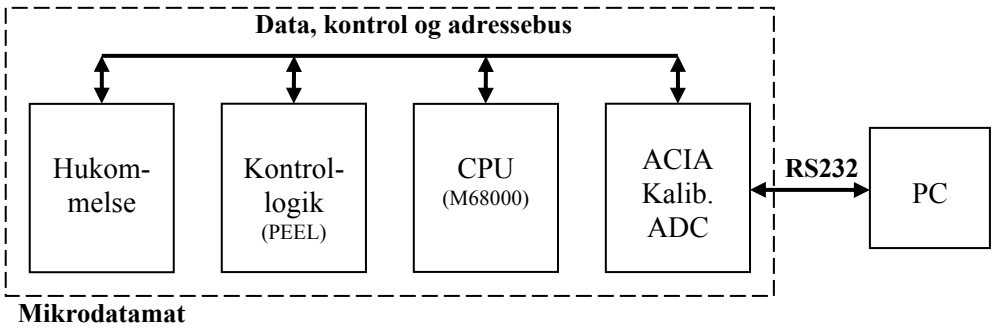
Da ½LSB svarer til 9,8mV kan det således konkluderes, at et s/h kredsløb i forbindelse med A/D-konverteringen ikke er nødvendig.

5.3 Delkonklusion og resultater

Test af det samlede A/D-konverteringssystem er blevet foretaget efter færdiggørelse af mikrodatamatens hardwaredel. A/D-konverteringssystemet er således blevet koblet sammen med mikrodatamaten i forbindelse med test. Denne test har vist at kendte signaler på A/D-konverteringssystemets indgange, giver et tilnærmelsesvis identisk konverteret udgangssignal. Den pågældende test er beskrevet i appendiks F.4.

6 Mikrodatamaten

For at omdanne de ubehandlede digitale signaler fra A/D-konverteren til muserapporter, som PC'en kan forstå, anvendes en mikrodatamat. Denne er opbygget af fire hovedblokke, der ses i figur 6.1. Hver blok vil blive beskrevet i dette kapitel.



Figur 6.1: Blokdiagram for mikrocomputeren

6.1 Overordnet beskrivelse

Det centrale i mikrodatamatsystemet er dets CPU[10]. For at CPU'en i første omgang kan afvikle programkode, er det påkrævet, at den har hukommelse til lagring af programmet, der skal afvikles, og til variable der anvendes. Disse to dele kan dog ikke uden videre forbindes. Dertil er påkrævet en adressedekoder, hvilket realiseres i kontrollogikken, der sørger for, ud fra CPU'ens udgange, at vælge hvilke kredse, der skal adresseres og indlæses data til/fra.
Systemets perifere enheder skal ligeledes styres af kontrollogikken. Periferienheder for det specifikke mikrodatamatsystem er en ACIA[11] til seriel kommunikation med PC'en, digitale modstande til kalibrering af EMG signalet og en analog-til-digital-konverter til digital konvertering af EMG-signalet. Ud over specifik software til muse-emulering kører TS2-monitoren, der er yderligere beskrevet i appendiks C, i baggrunden som operativsystem.

6.1.1 Overordnede krav

- CPU'en skal være hurtig nok til at behandle den nødvendige mængde data. Erfaringer viser at en clockfrekvens på 8MHz vil være passende. Den maksimale hastighed for den valgte M68HC000 er 12MHz.
- Størrelsen af ROM[12] skal være tilstrækkelig til at indeholde TS2-monitoren, hvilket svarer til et minimum på 16kByte, da TS2-monitoren fylder 11kByte.
- Størrelsen af RAM[13] skal være tilstrækkelig til at indeholde de nødvendige programmer (maks. 8kByte) samt programvariable (128 bytes). Desuden vil TS2-monitoren optage ca. 2kByte. Det forventes derfor, at 8kByte vil være tilstrækkeligt.
- Kontrollogikken skal kunne kontrollere, hvilke enheder der skal have adgang til bussen.

[10] Central Processing Unit
[11] Asynchronous Communication Interface Adapter
[12] Read Only Memory
[13] Random Access Memory

- De perifere enheder skal sætte mikrodatamaten i stand til serielt at kommunikere med en PC ved en hastighed på 1200 baud, da dette er krævet af musedriveren jf. appendiks B.1. Endvidere skal det vha. digitale modstande være muligt at styre forstærkningen.

6.2 CPU

Det er valgt at benytte en processor i mikrodatamaten af typen 68HC000 fra Motorola. CPU'en har en adressebus med en bredde på 24 bit, hvilket giver mulighed for håndtering af op til 16MB hukommelse. Endvidere har den en 16-bit databus til anvendelse ved dataoverførelse. Internt er CPU'en forsynet med 17 32-bit registre opdelt på følgende vis:

- 8 Dataregistre
- 7 Adresseregistre
- 1 User stack pointer
- 1 Program counter, der angiver hvilken del i programmet CPU'en er ved at afvikle.

Desuden findes også et 16-bit statusregister, der er nærmere beskrevet i appendiks G.1.2.

Eksternt kommunikerer CPU'en med omverdenen vha. sammenlagt 68 ben. Disse kan opdeles i 8 grupper, der ses herunder:

- 24-bit adressebus
- 16-bit databus
- Styring af synkrone enheder
- Styring af asynkrone enheder
- Funktionskontrol til udlæsning af CPU'ens tilstand
- Interruptkontrol
- Nødvendige signaler til drift af CPU'en herunder spændingsforsyning og clock.
- Styring af bus i forbindelse med flerprocessorsystemer.

En detaljeret beskrivelse af samtlige ben samt power-on-reset kredsløb og clock kredsløb kan ses i appendiks G.1.3.

6.3 Hukommelse

Udover CPU'ens interne hukommelse skal der bruges et ROM-lager til lagring af TS2-monitoren (se appendiks C). Det første CPU'en gør ved opstart er at læse på adresse $00 0000, hvorfor ROM'en skal placeres i denne adresse og opefter. Desuden skal der bruges et RAM-lager til lagring af eksterne programmer og programvariable.
Idet systemets databus har en bredde på 16 bit, og både RAM og ROM data ind- og udgange er 8-bit, er det nødvendigt at opdele hukommelsesområderne i både en øvre og en nedre del. Det betyder, at systemet rent fysisk skal have to RAM-kredse og to ROM-kredse.

6.3.1 ROM

Der er sat en række krav til ROM'en fra starten. Størrelsen skal mindst være 128 kbyte. Fra CPU'en er sat et krav til adresse tilgangstiden under en læsecyklus. Denne kan ses udregnet i appendiks G.2.1 til at skulle være mindre end 227,5ns.

Det er blevet valgt at anvende AM29F010B som systemets ROM. Tilgangstiden er i værste tilfælde 120ns, hvilket er indenfor det påkrævede. Størrelsen er 128 kbyte, hvilket også ligger indenfor kravene til ROM'en.

6.3.2 RAM

Grundlæggende findes der 2 typer RAM. Statisk og dynamisk RAM. Statisk RAM er opbygget således at det holder på data uden vedligehold, som den dynamiske RAM kræver. Dynamisk RAM kræver ligeledes at adresserne bliver multiplexet for korrekt brug. Der er således noget mere arbejde i at bruge dynamisk RAM end det statiske. Til gengæld er dynamisk RAM hurtigere, og kan lagre meget mere pr arealenhed end statisk.
Der vælges derfor statisk RAM, såfremt dette ikke vil sløve systemets hastighed ned. Der skal derfor undersøges hvilke krav der er til RAM'ens hastighed, og hvis statisk RAM kan leve op til disse vil denne type RAM blive valgt.
I appendiks G.2.2 kan tidskravene til hukommelsen ses udregnet. Det er fundet, at den statiske hukommelsesenhed SRM20100LC70 lever op til disse krav:

Parameter	Symbol	SRM20100LC70		Krav	Enhed
		Min	Max	Max	
Address access time	t_{acc}	-	*70*	*227,5*	*ns*
Input data setup time	t_{dw}	*30*	-	*180*	*ns*
Input data hold time	t_{dh}	-	*30*	*40*	*ns*

Tabel 6.1: Tider fra datablad for hukommelsesenhed

Hukommelsesenheden er på 128 kbyte.

6.4 Kontrollogik

For at kontrollere hvilke enheder der har adgang til bussen, er det nødvendigt at tildele hver hukommelsesdel samt periferienheder sit eget område indenfor det tilgængelige adresseområde. På figur 6.2 ses et såkaldt Memory Map, hvor inddelingen af enheder i adresseområdet fremgår.

Gruppe	Enhed	Adresse
00 0000h	TS2-monitor	00 0000h – 00 1200h
ROM		
03 FFFFh		
04 0000h	TS2-monitor	04 0000h – 04 FFFFh
RAM		
07 FFFFh		
08 0000h	ADC	08 0001h – 08 0007h (A19)
	Kalibrering	10 0000h (A20)
Periferienheder		
	ACIA	80 0001h – 80 0003h (A23)
100 0000h		

Figur 6.2: Memory map

Placeringen af RAM-, ROM- og ACIA er på forhånd fastlagt af den anvendte TS2-monitor. Som det fremgår af figuren, er det ikke hele adresseområdet, der er anvendt, hvilket gør adressedekodningen væsentlig mere simpel, end hvis det modsatte var tilfældet.

6.4.1 Adressedekodning

Det er valgt at implementere adressedekodningen i den samme PEEL-kreds, som bliver anvendt i forbindelse med AD-konverteren, da der var ubrugte ressourcer i denne.

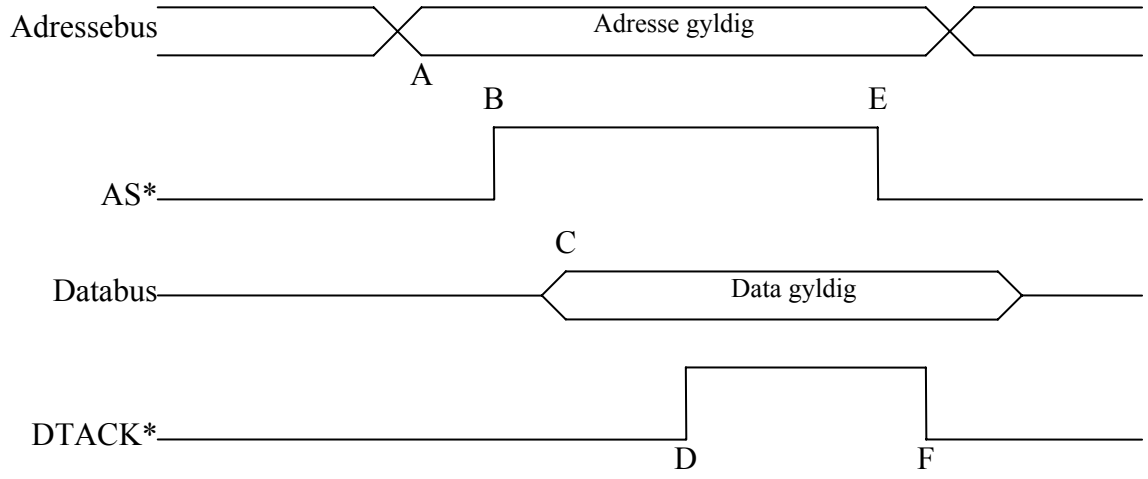
Indgange						Udgange						
A18	**A19**	**A20**	**A23**	**AS***	**VMA***	**VPA***	**ROM***	**RAM***	**ACIA***	**ADC**	**KAL***	**DTACK***
0	0	0	0	1	X	0	1	0	0	0	0	1
1	0	0	0	1	X	0	0	1	0	0	0	1
0	0	0	1	1	1	1	0	0	1	0	0	1
0	1	0	0	1	X	0	0	0	0	1	0	1
0	0	1	0	1	0	0	0	0	0	0	1	1

Tabel 6.2: Sandhedstabel for adressedekodning i PEEL-kreds

For at forstå tabel 6.2 til fulde, er det nødvendigt at beskrive begreberne DTACK*, AS*, VMA* og VPA*, der anvendes til henholdsvis asynkron og synkron busadgang. En overordnet beskrivelse af signalerne findes ligeledes i appendiks G.3.

Asynkron buskontrol

DTACK* er en indgang til CPU'en genereret, gennem kontrollogikken, af den enhed CPU'en tilgår. DTACK* går aktiv høj, betyder det, at enheden er klar til at blive tilgået af CPU'en. Hvis ikke et DTACK* signal genereres, vil CPU'en generere wait-states indtil dette sker. DTACK* kan derfor betragtes som en bekræftelse fra de eksterne enheder, der er tilsluttet bussen. Desuden vil AS* gå aktiv høj, når der er en gyldig adresse på adressebussen, hvilket fremgår af figur 6.3.



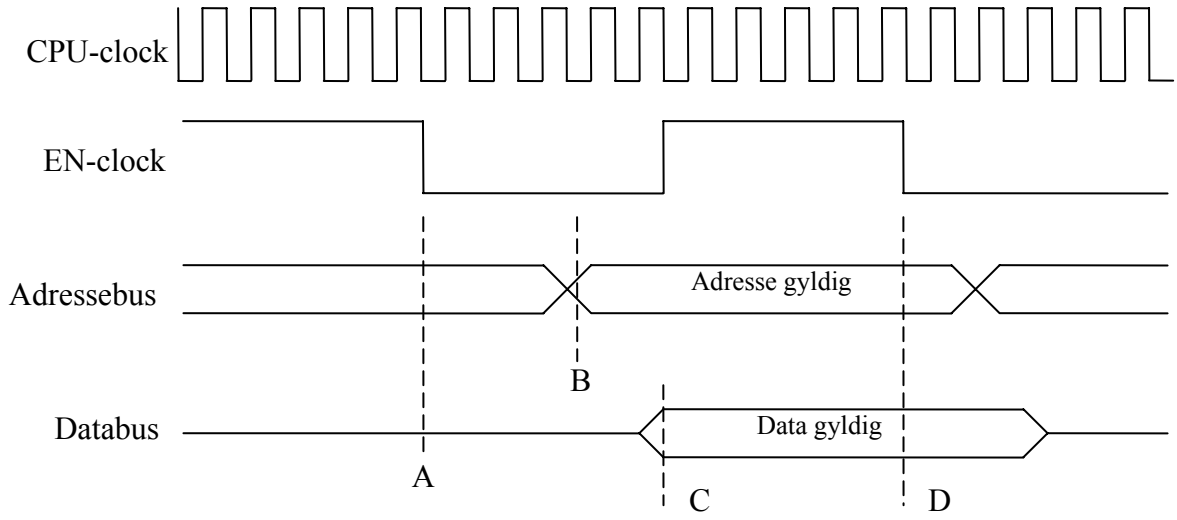
Figur 6.3: Asynkron buskontrol

På tidspunktet A er adressebussen sat, hvilket sætter AS* aktiv høj (B). Enheden der tilgås vil detektere at AS* er høj og efterfølgende ligge data på databussen (C). Kontrollogikken vil nu sætte DTACK* aktiv høj, således at CPU'en ved, at indholdet af databussen er gyldigt. Efterfølgende

sættes AS* lav for at informere enheden om at dataene er læst (E). Til slut sættes DTACK* lav, som indikation på at cyklussen er gennemført.

Synkron buskontrol

VPA* og VMA* anvendes i forbindelse med synkron busadgang, hvilket kun er nødvendigt, hvis der bruges ældre perifere enheder såsom ACIA 6850. Det synkrone består i at CPU'en og den perifere enhed udfører handlinger baseret på den samme clock. Denne clock består i CPU'ens EN-udgang, der som tidligere beskrevet, har en frekvens på 1/10 af CPU'ens clock. Clocken er tilsluttet ACIA'ens E-indgang.
Ønsker en enhed synkron adgang til bussen sætter den, vha. kontrollogikken, VPA* indgangen i CPU'en logisk høj. Dette medvirker, at CPU'en igangsætter en synkron buscyklus, der ideelt set foregår som angivet på figur 6.4.



Figur 6.4: Synkron buskontrol

Til tidspunktet A begynder en læsecyklus, hvilket vil få CPU'en til at aktivere adressebussen (B). Efterfølgende vil enheden, der tilgås, lægge data klar på databussen (C), som ved næste faldende clock vil blive læst af CPU'en (D).
En mere specifik gennemgang af synkron buskontrol fremgår af appendiks G.3.1.

Eksempel på PEEL-kode

Til selektering af ROM'en anvendes koden som fremgår af nedenstående eksempel:

```
CS_ROM.COM = !(A18 + A19 + A20 + A23) & !AS;
CS_ROM.OE = 1;
```

Syntaks:
!: NOT
&: AND
+: OR
.COM: Angiver at udgangen er kombinatorisk.
.OE: Aktiverer udgangen.

Af det pågældende eksempel fremgår det, at ROM'en kun vil blive selekteret i det tilfælde at hverken RAM (A18), ADC (A19), digitale modstande (A20) eller ACIA (A23) er selekteret. Desuden skal AS* være aktiv høj. Resten af koden kan ses i appendiks G.3.2.

Selektering af øvre og nedre del af hukommelsesområdet sker udenfor PEEL-kredsen vha. henholdsvis UDS og LDS, hvilket fremgår af diagrammet i appendiks K.5.

6.4.2 Interruptkontrol

Interrupts anvendes, som navnet antyder, til at afbryde CPU'en i det den er ved at udføre. Når et interrupt er udført, vender CPU'en tilbage til den tilstand og det sted, den var inden interruptet. Interrupts generes af perifere enheder gennem CPU'ens tre interruptindgange. Idet der kun vil blive anvendt tre interrupts udover NMI-switchen (beskrevet på følgende side), er der ikke behov for en decideret interrupt dekodning. I tabel 6.3 ses en oversigt over de anvendte interrupts i mikrodatamaten.

I2	I1	I0	#	Enhed
1	1	1	7	NMI-switch
1	1	0	6	-
1	0	1	5	-
1	0	0	4	ACIA
0	1	1	3	-
0	1	0	2	ADC2 (3+4)
0	0	1	1	ADC1 (1+2)

Tabel 6.3: Interruptoversigt

Des højere nummer en interrupt har des højere prioritet. Dette betyder, at NMI-switchen har højeste prioritet, og derfor vil den tilhørende interruptrutine altid blive udført uanset CPU'ens tilstand (undtagen fejl). Interrupt 7 er den eneste, der ikke kan deaktiveres.

I appendiks G.3.3 kan ses en beskrivelse af en interruptscyklus.

Der findes to forskellige typer interrupts: Vektoriseret og autovektoriseret. Ved et vektoriseret interrupt vil en enhed, når den interrupter CPU'en, gennem de første otte bit på databussen, identificere sig. Dette betyder, at den perifere enhed gennem databussen angiver, hvilken interruptrutine der skal udføres. Ved et autovektoriseret interrupt vil CPU'en internt generere den pågældende interrupt uafhængigt af indholdet af databussen. Flow charts for de to interrupttyper kan ses i appendiks G.3.
I det pågældende system vil der kun blive anvendt autovektoriserede interrupts, idet de anvendte perifere enheder ikke er i stand til at forsyne databussen med de nødvendige interruptvektorer.

Ved et interrupt vil CPU'ens funktionsudgange alle være høje (jf. tabel G.1), hvilket vil blive anvendt til generering af VPA-signalet til CPU'en jf. tabel 6.4.

Indgange					Udgange
FC2	FC1	FC0	AS*	VPA*-PEEL	VPA*-CPU
1	1	1	1	1	1
0	X	X	X	X	0
X	0	X	X	X	0
X	X	0	X	X	0
X	X	X	0	X	0
X	X	X	X	0	0

Tabel 6.4: Generering af VPA-signal til CPU

VPA-signalet aktiverer, som beskrevet i appendiks G.3, enten et autovektoriseret interrupt eller synkron busadgang.

NMI[14]-switch

Der er i TS2-monitoren indbygget understøttelse af en såkaldt NMI-switch, der giver mulighed for at standse eksekveringen af kode i CPU'en.
Selve kontakten udløser et interrupt 7, der ikke kan maskes, og er stort set opbygget på samme måde som power on reset kredsløbet, hvilket fremgår af diagrammet i appendiks K.5.

6.5 Perifere enheder

De perifere enheder, ACIA, ADC og digitale modstande, sætter CPU'en i stand at kommunikere med omverdenen på forskellig vis. ADC'en vil ikke blive yderligere beskrevet her, da dette er gjort i kapitel 5.

6.5.1 ACIA

Formålet med ACIA'en er at konvertere mellem parallelle data fra CPU'en til serielle data som eksempelvis kan sendes til eller modtages fra en PC. Det specifikke valg af kreds er faldet på ACIA 6850, da denne er forholdsvis simpel og lever op til de stillede krav. ACIA 6850 er bl.a. kendetegnet ved, at den kører synkront med CPU'en, hvilket er beskrevet i afsnit G.3.1.

Registre

ACIA'en indeholder fire registre: Status register, kontrol register, samt henholdsvis et sende og et modtage register, der indeholder de data, der enten skal sendes eller er modtaget. Status- og kontrol registret er yderligere beskrevet i appendiks G.4.1. Selektering af de fire registre sker vha. en kombination af RS og R/W* indgangene i ACIA'en jf. tabel 6.5.

[14] None Maskable Interrupt

RS	R/W*	Register type	Register funktion
0	1	Skriv	Kontrol register
0	0	Læs	Status register
1	1	Skriv	Sende register
1	0	Læs	Modtage register

*Tabel 6.5: Selektering af registre vha. RS og R/W**

Ind-/udgange

En oversigt samt en generel beskrivelse af ACIA'ens ben kan ses i appendiks G.4.2.
Selektering af ACIA'en sker vha. af en enkelt indgang, CS2*, der er tilsluttet til PEEL-kredsen. ACIA'en er synkroniseret med CPU'en via EN-udgangen på CPU'en og E-indgangen på ACIA'en, der som tidligere nævnt har en clockfrekvens på 1/10 af CPU'ens clock. Da det ønskes, at ACIA'en skal sende og modtage serielle data med samme hastighed, kan de to clock-indgange i ACIA'en umiddelbart begge tilsluttes baudgeneratoren. Der findes på ACIA'en to indgange (CTS* og DCD*) samt en udgang (RTS*), der alle er en del af RS232 standarden. I dette projekt ønskes kun CTS* anvendt.
I ACIA 6850 findes en enkelt interrupt udgang, der kan konfigureres vha. kontrolregistret og desuden aflæses gennem statusregistret. ACIA'en kan sættes til at interrupte både ved modtaget samt afsendt data, hvor begge dele benyttes i det aktuelle system. Mere herom i afsnit 7.2.
Endvidere har ACIA'en en TTL-indgang samt -udgang til modtagelse og afsending af data, hvilket kan ske simultant (full duplex). ACIA'en er koblet til databussens 8 nedre bit.

6.5.2 MAX232

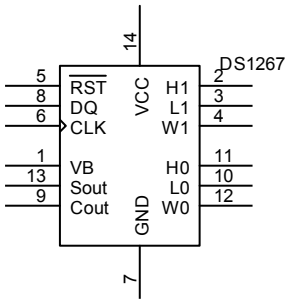
Det serielle signal fra ACIA'en er givet ved TTL-niveauer, hvorfor det er nødvendigt at omdanne disse til RS232-niveauer, således at PC'ens serielport kan modtage dataene. Dette gøres med en simpel kreds, MAX232, der blot behøver 5V forsyning og internt laver de nødvendige spændingsniveauer. Opkoblingen af kredsen kan ses i appendiks K.5.

6.5.3 Baudgenerator

TS2-monitoren indstiller ACIA'en til en deling af indgangsclocken med 16, hvilket svarer til indgangsclock på 19200 baud, hvis der skal opnås en sende-/modtagehastighed på 1200 baud, som er påkrævet af musestandarden. Denne clockfrekvens genereres af en såkaldt baudgenerator af typen MC14411, der er i stand til at genere TTL-signaler med en clockfrekvens på mellem 75Hz og 1,843MHz ved en dutycycle på 50%. Opkoblingen af kredsen fremgår af diagrammet i appendiks K.5.

6.5.4 Digitale modstande

Det er valgt at benytte digitale modstande af typen DS1267-100 [CD\Datablade\DS1267.pdf] til kalibrering af forstærkningen. Denne kreds er valgt, idet den indeholder to digitale potentiometre, har en opløsning på 8bit og er i stand til at yde en modstand på mellem 400Ω og $100k\Omega$ jf. afsnit 3.2.
Kredsen DS1267-100 er illustreret i figur 6.5.



Figur 6.5: Diagram over DS1267

Inden uddybningen af kredsens virkemåde følger først en beskrivelse af kredsens forskellige input og output.

Input

Kredsen har følgende 6 input ben:

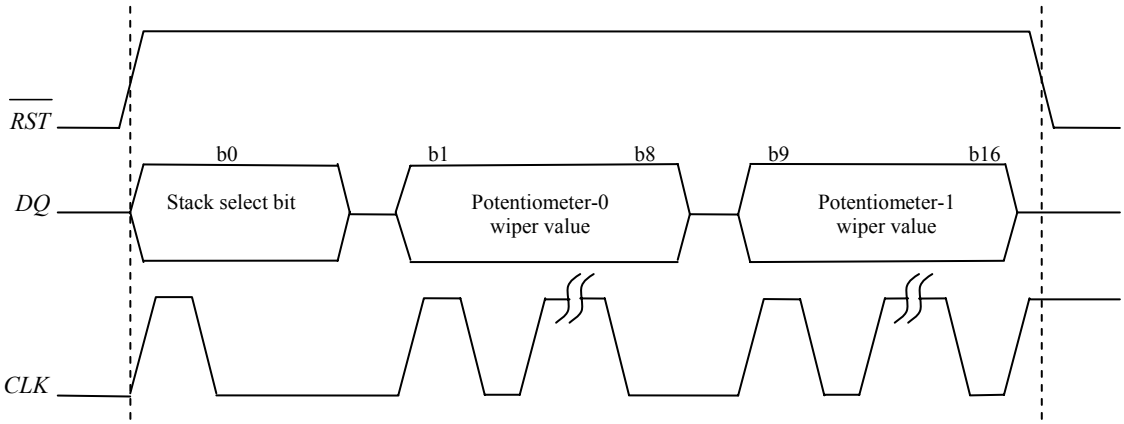
- *VB* sætter kredsens bias spænding, og sættes i dette tilfælde til -Vcc således spændingen over de digitale modstande kan ligge mellem -Vcc og Vcc.
- $\overline{RST}$ bestemmer hvornår kredsen indlæser nye data i kredsens interne skifteregister. Når denne går høj kan kommunikation med kredsen påbegyndes, og når denne igen går lav tages de modtagne data i brug i de digitale modstande.
- *CLK* benyttes til at synkronisere indlæsning af input og vil i dette tilfælde blive genereret i mikrodatamatens program.
- *DQ* er kredsens serielle datainput. Dette ben benyttes til at indlæse de data der skal bestemme de digitale modstandes værdi.
- *GND* kobles til digital GND.
- *Vcc* kobles til 5V digital spændingsforsyning.

Output

- *Sout* og *Cout* benyttes i det tilfælde at flere kredse kobles sammen, og vil således ikke blive benyttet i det aktuelle tilfælde.
- *W0* og *W1* er potentiometrenes variable ben, idet de indstilles på baggrund af kredsens serielle datainput, når $\overline{RST}$ går lav.
- *L0* og *L1* er potentiometrenes lave ender. Spændingsfaldet over de to variable modstande vil ved implementering således ligge over L0 og W0 samt L1 og W1.
- *H0* og *H1* er potentiometrenes høje ender. Disse vil ved implementering ikke blive koblet til noget.

Virkemåde for DS1267

Virkemåden for DS1267 er illustreret i figur 6.6



Figur 6.6: Timingsdiagram over indlæsning af data til DS1267[CD:\Datablade\DA1267.pdf s.10]

Indlæsning af data på DS1267 starter ved at $\overline{RST}$ sættes høj. Ved første lav til høj ændring i klokken, indlæses bit 0, stak bitten, i kredsens skifteregister. Denne bit afgør hvorvidt potentiometrene kører enkeltvis eller er stakket sammen i et 9 bits potentiometer. Da potentiometrene skal fungere enkeltvis sættes stak bitten til 0.
Ved de efterfølgende lav til høj ændringer i klokken, indlæses bit 1 til 16 enkeltvis, 8 bit til hvert potentiometer.
Når alle 17 bit er indlæst sættes $\overline{RST}$ lav igen, hvorved data i skifteregistreret indlæses i potentiometrene, hvorved W0 og W1 værdierne sættes.

Implementering

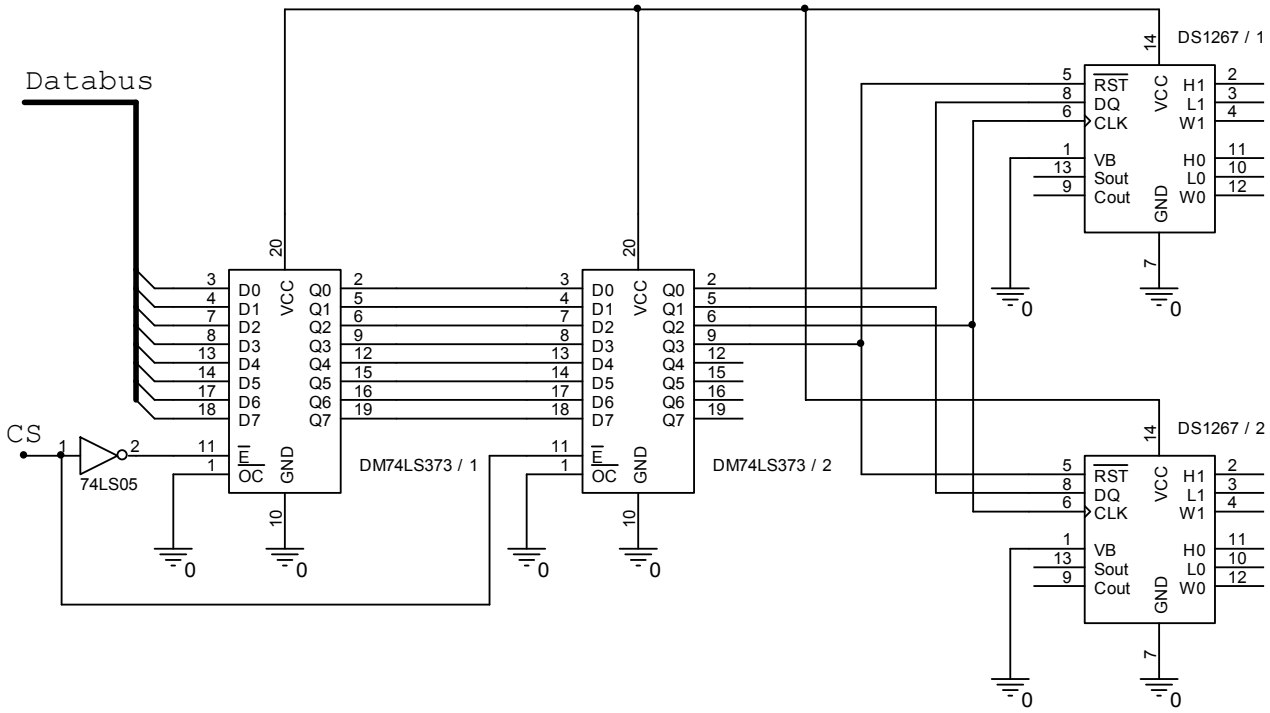
Da de fire kanaler med EMG-signaler alle skal forstærkes, er det nødvendigt at benytte to DS1267 kredse. Hver kreds skal modtage input svarende til figur 6.6. $\overline{RST}$ og CLK er identiske for de to kredse, hvorimod hver kreds naturligvis har forskellige datainput. Disse fire input genereres i mikrodatamatens programmel jf. appendiks I, og har adgang til de digitale modstande gennem databussen. Da mikrodatamaten som minimum kan sende én byte, vil de fire input bestå af de fire mindst betydende bit i den afsendte byte, jf. tabel 6.6.

Bit på databussen	Ben på DS1267 kredsene
b0	DQ på kreds 1
b1	DQ på kreds 2
b3	Fælles CLK
b4	Fælles $\overline{RST}$
b5-b7	Ingen forbindelse

Tabel 6.6: Databussens adgang til de to DS1267 kredse

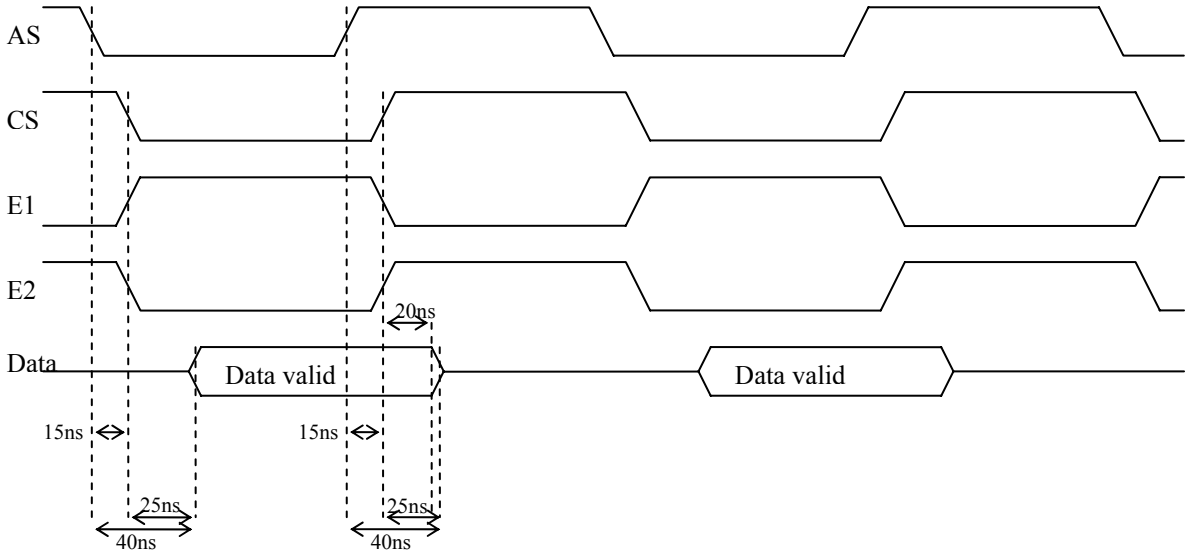
Da hver DS1267 kreds har behov for 17 databit serielt indlæst og hver databit indlæses ved en lav til høj ændring i klokken CLK, er det nødvendigt at hver databit indlæses to på hinanden følgende gange, således klokken først er lav og derefter høj. Dette betyder, at der skal sendes 34 pakker (byte) plus en nulpakke, der sætter $\overline{RST}$ lav, over databussen.
Som førnævnt skal $\overline{RST}$ holdes høj under dataindlæsningen. Dette vil ikke være muligt, hvis DS1267 kredsene kobles direkte til databussen, idet databussen slipper forbindelsen til kredsene efter hver enkelt indlæst byte.

Det er således fundet nødvendigt at indsætte en latch-kreds mellem databussen og DS1267 kredsene. Det er valgt at benytte kredsen DM74LS373 [CD:\Datablade\ DM74LS373.pdf], da denne indeholder 8 latches og er forholdsvis enkelt at implementere. Dette løser dog ikke fuldstændigt problemet med at holde $\overline{RST}$ høj, hvorfor det er valgt at indsætte to DM74LS373 kredse mellem databussen og DS1267 kredsene. Figur 6.7 illustrerer hvorledes dette er implementeret.



Figur 6.7: Diagram over implementeringen af DS1267 kredsene og DM74LS373 kredsene

Som vist på figuren svæver de øverste fire bit, af den tilsendte byte, da blot de nederste fire bit benyttes til at styre indstillingen af potentiometret.
Grunden til at én latch-kreds ikke er tilstrækkeligt i det aktuelle system er, at fra mikrodatamaten sætter AS lav går der 40 ns før data ligger klar på databussen [Micro s.236]. Når adressedekoderen modtager AS lav, går der 15ns før CS til den aktuelle kreds går lav [CD:\Datablade\PA7540DS.pdf s.7]. Der er således 25ns, hvor den aktuelle kreds er selekteret, men der ikke ligger korrekt data på databussen. Under normale omstændigheder ville dette ikke udgøre et problem, men da $\overline{RST}$ *skal* holdes høj under indlæsningen af alle 35 byte, ville denne periode, på 25ns, forårsage ukorrekte data i de digitale modstande.
Dette problem løses ved at sætte to latch-kredse i serie og forbinde latch-kreds nr. 1's enable ben til adressedekoderens CS gennem en inverter, og latch-kreds nr. 2's enable ben direkte til adressedekoderens CS. Latch-kredsene enables dermed skiftevis således, at ukorrekte data ikke har adgang til DS1267 kredsene. Dette kræver dog at data på databussen, ved lav til høj ændring på CS, ligger tilstrækkelig længe til at latch-kreds nr. 1 kan nå at låse data fast inden disse fjernes. Dataene på databussen vil ligeledes blive liggende 25ns efter CS til latch-kreds nr.1 er gået høj. Dette er tilstrækkeligt, idet kredsen har en data hold time på 20ns. Når $\overline{E}$ benet på latch-kredsen er højt er data på input og output siden identisk, medens data på output siden er låst fast når $\overline{E}$ er lav.
$\overline{OC}$ benet sættes til fast lavt niveau således, at output siden på intet tidspunkt bliver høj-impedant.
De to latch-kredses samlede virkemådes timingdiagram er illustreret i figur 6.8.



Figur 6.8: Timingdiagram over de to DM74LS373 kredse

6.6 Delkonklusion

Følgende krav er opstillet til mikrodatamaten:

- CPU'en skal have en clockfrekvens på 8MHz.
- Størrelsen af ROM skal minimum være på 16kByte.
- Størrelsen af RAM skal minimum være på 8kByte.
- Kontrollogikken skal kunne kontrollere, hvilke enheder der skal have adgang til bussen.
- De perifere enheder skal sætte mikrodatamaten i stand til serielt at kommunikere med en PC ved en hastighed på 1200 baud. Desuden skal det være muligt at styre forstærkningen vha. digitale modstande.

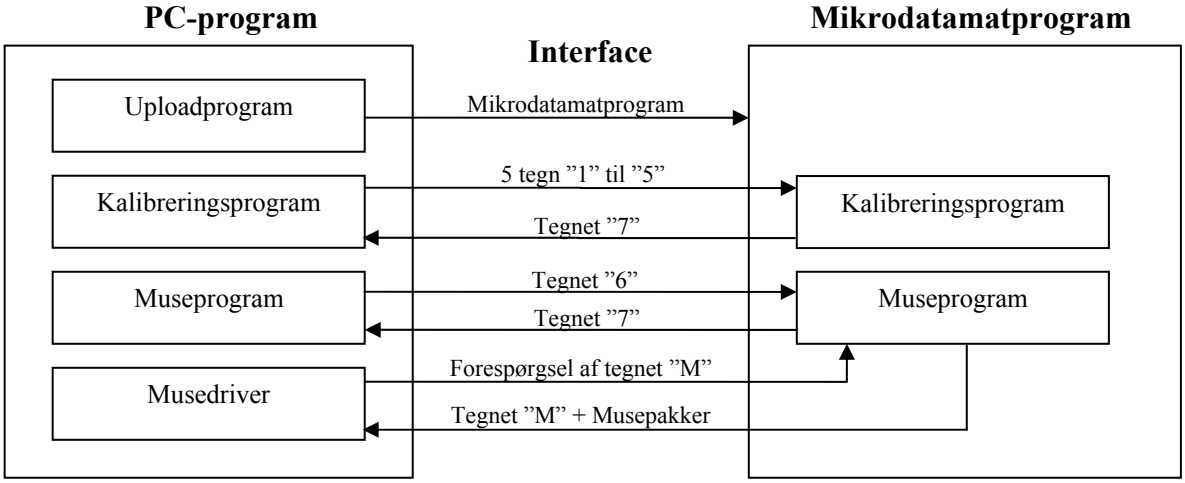
Den endelige mikrodatamat har nu følgende specifikationer:

- CPU'ens clockfrekvens er på 8MHz.
- Der er 2x128kByte ROM samt 2x128kByte RAM.
- Kontrollogikken fungerer efter hensigten.
- Mikrodatamaten er i stand til at kommunikere med PC'en ved en hastighed på 1200 baud. Der er i det endelige system ikke mulighed for at regulere forstærkningen vha. digitale modstande.

Som det fremgår lever mikrodatamaten op til de stillede krav med undtagelse af digital regulering af forstærkningen. Dette blev afhjulpet ved at erstatte de digitale modstande med trimmemodstande, hvilket betyder, at brugeren manuelt skal indstille forstærkningen. Grunden til problemerne med de digitale modstande er sandsynligvis timingsproblemer. Test af systemet er udført og viser, at systemet fungerer efter hensigten, med undtagelse af de digitale modstande, jf. appendiks J.

7 Software

Den EMG-styrede mus's softwaredel består af to dele; et PC-program og et program til mikrodatamaten.
Denne opdeling foretages, idet mikrodatamatprogrammet skal gennemføre alle aktuelle udregninger og sende musepakker til PC'en. PC-programmet skal blot virke som en brugerflade til mikrodatamatprogrammet, når dette først er uploaded i mikrodatamaten.
Softwarens overordnede virkemåde er beskrevet i kravsspecifikationen, medens en udførlig beskrivelse af softwaren følger i afsnittene i dette kapitel. Inden afsnittene, omhandlende softwaren til henholdsvis PC'en og mikrodatamaten, vil der her blive foretaget en kort gennemgang af interfacet mellem disse.
Dette interface er illustreret i figur 7.1.



Figur 7.1: Interface mellem PC-programmet og mikrodatamatprogrammet

Via uploadprogrammet på PC'en uploades mikrodatamatprogrammet til mikrodatamaten, hvilket er det første interface mellem de to systemer. Dette interface er gjort muligt af TS2-monitoren, der er brændt ned i mikrodatamatens ROM-kredse. Mikrodatamaten vil naturligvis benytte TS2-monitoren konstant, men da dette betragtes som indforstået, vil TS2-monitoren ikke blive nævnt i forbindelse med de øvrige interfaces.
Endvidere findes et interface mellem kalibreringsprogrammerne på PC'en og mikrodatamaten. Dette interface består bl.a. af et af fem tegn, "1" til "5", afsendt fra PC'en, der tilkendegiver overfor mikrodatamatens kalibreringsprogram, hvilken kanal brugeren ønsker kalibreret, samt hvorvidt offset- og tærskelværdier skal bestemmes. Ved aktivering af mikrodatamatens kalibreringsprogram, og ved gennemført kalibrering, afsendes tegnet "7" til PC'ens kalibreringsprogram som bekræftelse på henholdsvis påbegyndt og gennemført kalibrering. Ved opstart af museprogrammet i mikrodatamaten afsendes ligeledes tegnet "7" som bekræftelse på, at dette er sket.
Der benyttes desuden et interface fra PC'ens museprogram til mikrodatamatens museprogram. Dette interface består af tegnet "6", der tilkendegiver overfor mikrodatamaten, at dennes museprogram skal opstartes, hvorved tegnet "7" sendes fra mikrodatamatens museprogram som bekræftelse på, at dette er sket.
Endelig benyttes et interface mellem mikrodatamatens museprogram og PC'ens musedriver. Dette interface består af en initierende forespørgsel fra musedriveren på tegnet "M". Dette besvares af

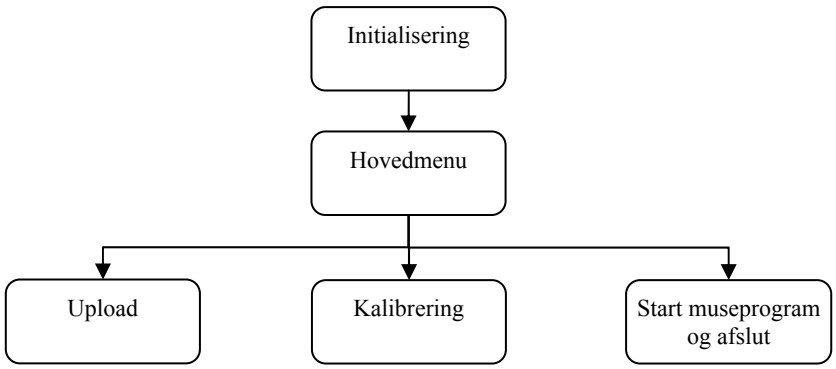
mikrodatamatens museprogram med netop afsendelse af tegnet ”M”, hvilket indikerer overfor musedriveren, at en mus er tilsluttet COM1 porten. Dette følges af en kontinuerlig strøm af musepakker til PC’ens musedriver.

7.1 PC software

Formålet med PC softwaren er at skabe et interface mellem brugeren og mikrodatamaten. Det skal således være dette interface, der giver brugeren adgang til mikrodatamatens forskellige funktioner. Det er valgt at implementere denne software i programmeringssproget C.
Gennem kravspecifikationen er der blevet opstillet en række krav, der lægger grundlaget for PC softwarens design. Kravene til PC softwaren er opstillet nedenfor:

- Skal designes til et Windows® styresystem
- Skal have en simpel brugergrænseflade
- Skal bestå af en menu, der giver mulighed for at:
    - Uploade assembler softwaren til mikrodatamaten
    - Styre kalibreringen de 4 kanaler enkeltvis på mikrodatamaten, samt instruere brugeren i anvendelse af dette
    - Bestemme offset- og tærskelværdier
    - Starte museprogrammet på mikrodatamaten
    - Afslutte C-programmets brug af COM1 porten
    - Afslutte C-programmet

På baggrund af disse krav er det valgt at opbygge PC softwaren som vist på figur 7.2



Figur 7.2: PC softwaren overordnede opbygning

De enkelte blokke i figur 7.2 vil herefter blive beskrevet, og på baggrund af denne beskrivelse er den endelige kildekode designet. Kildekoden kan ses i appendiks H og indeholder endvidere en mere specifik forklaring på de benyttede kommandoer.

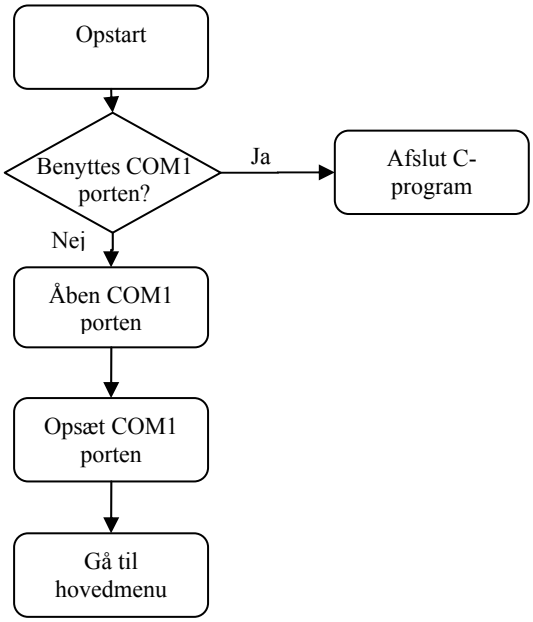
7.1.1 Initialisering/hovedmenu

Ved programstart er det nødvendigt at initialisere den serielle COM1 port, således at denne er i stand til at kommunikere med mikrodatamaten. TS2-monitoren og MS Mouse Driver kræver, at COM1 porten skal sættes til følgende indstillinger:

- Hastighed: 1200 baud
- Antal databits: 8

- Paritetsbit: Ingen
- Stopbit: 1

Flowdiagrammet for funktionen ”Initialisering” ses i figur 7.3.

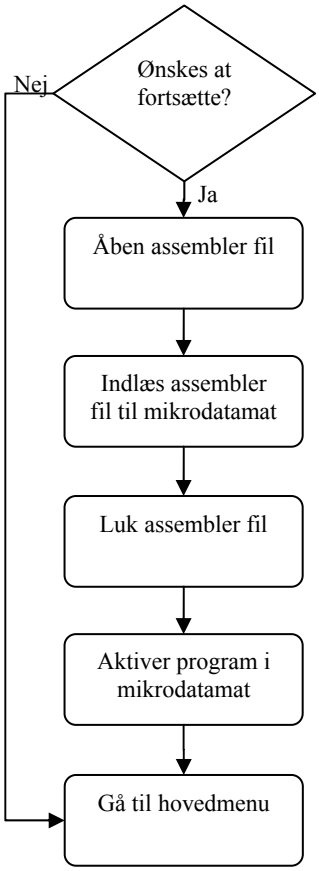


Figur 7.3: Flowdiagrammet for initialiseringsfunktionen

Efter opstart af C-programmet testes for hvorvidt COM1 porten allerede bliver benyttet eller ej. Benyttes den ikke åbner C-programmet COM1 porten, hvorefter denne opsættes. COM1 porten er således initialiseret, hvorefter C-programmets hovedmenu kaldes.
Benyttes COM1 porten i forvejen afsluttes C-programmet.

7.1.2 Upload

Efter initialiseringen er det nødvendigt som det første at vælge ”Upload” i hovedmenuen, idet denne funktion uploader og starter assembler programmet i mikrodatamatens RAM. Flowdiagrammet for funktionen ”Upload” ses i figur 7.4.

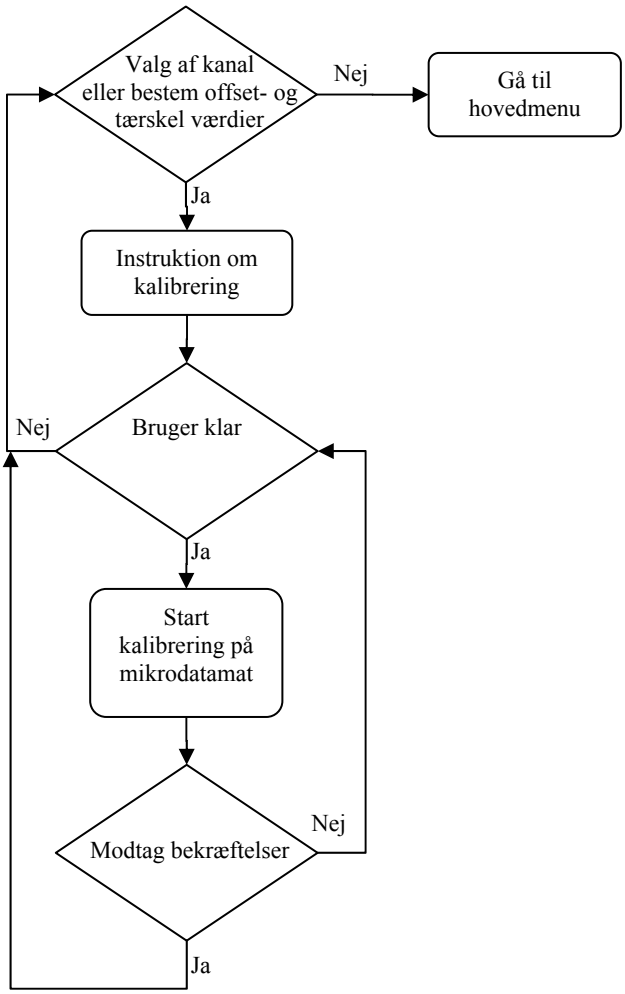


Figur 7.4: Flowdiagrammet for uploadfunktionen

Når først "Upload" kommandoen er valgt aktiveres upload-funktionen, hvorefter brugeren skal angive, hvorvidt det ønskes at fortsætte eller gå tilbage til hovedmenuen. Vælges det at fortsætte åbner C-programmet filen program.rec, hvorefter indholdet uploades i mikrodatamatens RAM. C-programmet lukker herefter filen program.rec og aktiverer derpå programmet beliggende i mikrodatamaten med kommandoen JUMP 41000, således at assemblerprogrammet nu er aktivt. Efter uploading og aktivering af assemblerprogrammet returneres til hovedmenuen.

7.1.3 Kalibrering

Kalibreringen foretages med det formål, at brugeren kan indstille musemarkørens følsomhed ved regulering af forstærkningen styret af de digitale modstande. Kalibreringen er nødvendig, idet det skal være muligt at forskellige brugere kan benytte den EMG-styrede mus. Desuden giver det mulighed for, at brugeren selv kan bestemme, hvilke muskler der ønskes anvendt, da forskellige muskler giver forskelle i størrelsen af EMG-signaler. Flowdiagrammet for funktionen "Kalibrering" ses i figur 7.5.

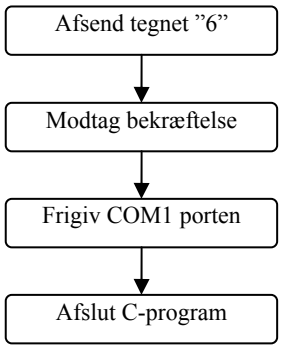


Figur 7.5: Flowdiagrammet for kalibreringsfunktionen

Når ”Kalibrering” er valgt aktiveres kalibreringsfunktionen, hvorefter brugeren skal angive, hvilken kanal der ønskes kalibreret, om det ønskes at bestemme offset- og tærskelværdier eller om det ønskes blot at gå tilbage til hovedmenuen. Vælges det at kalibrere en kanal eller bestemme offset- og tærskelværdier, vil brugeren modtage oplysninger forklarende, hvorledes dette vil foregå. Herefter skal brugeren angive, hvorvidt han/hun er klar til at fortsætte eller ej. Er brugeren ikke klar returneres til undermenuen, hvor brugeren igen kan vælge at kalibrere en kanal eller gå tilbage til hovedmenuen. Ønsker brugeren derimod at fortsætte, startes kalibreringsprogrammet på mikrodatamaten. Herefter venter C-programmet på bekræftelser fra mikrodatamaten om først påbegyndt kalibrering og derefter endt kalibrering. Modtager C-programmet ikke disse bekræftelser returneres til stedet, hvor brugeren skal indikere om han/hun vil forsøge med en ny kalibrering på samme kanal. Modtager C-programmet bekræftelserne fra mikrodatamaten, returneres til undermenuen, hvor brugeren kan vælge at kalibrere endnu en kanal eller gå tilbage til hovedmenuen.

7.1.4 Start museprogram og afslut

Efter endt indlæsning af assembler programmet og evt. kalibrering skal selve mikrodatamatens museprogram aktiveres. Dette gøres ved at vælge ”Start museprogram” i hovedmenuen. Flowdiagrammet for opstart af museprogram, funktionen ”Opstart”, ses i figur 7.6.



Figur 7.6: Flowdiagrammet for start funktionen ”Opstart”

Når ”Start museprogram og afslut” kommandoen er valgt, vil C-programmet som det første afsende tegnet ”6” til mikrodatamaten, hvilket indikerer, at mikrodatamaten skal aktivere sit eget museprogram. C-programmet vil så afvente bekræftelse fra mikrodatamaten, på at denne har aktiveret museprogrammet. Modtages denne bekræftelse ikke returneres til undermenuen ”Start museprogram og afslut”. Modtages denne bekræftelse derimod frigives COM1 porten, idet denne skal benyttes af MS Mouse Driveren, hvorefter C-programmet afsluttes. Mikrodatamaten vil nu afvente, at CTS går lav, hvorefter et antal ”M”’er afsendes. I Windows® skal brugeren manuelt søge efter ny hardware, hvilket medfører at CTS går lav og mikrodatamaten afsender sine ”M”’er. Windows® installerer nu den nye mus, og mikrodatamaten begynder at afsende musepakker.

7.2 Program til mikrodatamaten

Formålet med programmet til mikrodatamaten er, på baggrund af EMG-signaler, at emulere en computermus. Det er valgt at implementere dette program i assemblerkode.
Gennem kravspecifikationen er der opstillet en række krav, der lægger grundlaget for assemblerkodens design. Kravene til assemblerkoden er opstillet nedenfor:

- Kanal 1 skal udgøre en markørændring til højre
- Kanal 2 skal udgøre en markørændring til venstre
- Kanal 3 skal udgøre en markørændring op
- Kanal 4 skal udgøre en markørændring ned
- Ved samtidig aktivering af kanal 1 og 2 skal foretages venstreklik
- Ved samtidig aktivering af kanal 3 og 4 skal foretages højreklik
- Skal indeholde et kalibreringsprogram til kalibrering af kanalerne
- Skal indeholde et kalibreringsprogram til bestemmelse af offset- og tærskelværdierne
- Modtagne EMG-signaler under de bestemte tærskelværdien skal sættes til værdien 0, således, at svage EMG-signaler ikke forårsager uønskede klik og bevægelser
- Musepakkerne skal sendes i overensstemmelse med MS Mouse Driver (jf. appendiks B)
- Assembler koden skal servicere interrupts.

På baggrund af disse krav er det valgt at designe assemblerkode for følgende punkter:

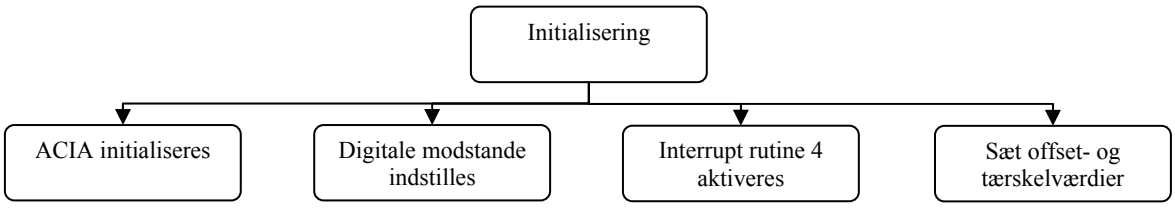
- Initialisering
- Interrupt rutine 1 og 2 (Interrupts fra A/D-konverteringen)
- Interrupt rutine 4 (Interrupt fra ACIA’en)

- Kalibrering af offset- og tærskelværdier
- Kalibrering af valgt kanal
- Museopstart
- Styring af musepakker
- Behandling af data

Disse punkter vil blive gennemgået enkeltvis og vil tilsammen udgøre mikrodatamatens programmel. Dette programmel vil i størst mulige udstrækning være interruptstyret. Kildekoden til mikrodatamatens programmel kan ses i appendiks I.

7.2.1 Initialisering

Initialiseringen bliver aktiveret af PC'en umiddelbart efter, at denne har uploaded assemblerkoden i mikrodatamatens RAM. Initialiseringens overordnede formål er således at klargøre mikrodatamaten til at kunne køre det assemblerkode, der er nødvendigt for, at mikrodatamaten kan generere musepakker og kommunikere med C-programmet. Figur 7.7 illustrerer dette.



Figur 7.7: Funktionen "Initialisering"s arbejdsopgaver

ACIA'ens kontrolregister sættes til kun at interrupte ved modtaget data, idet interruptrutine 4 på dette tidspunkt kun ønskes benyttet ved modtaget data på ACIA'en. I de digitale modstande indlæses den binære reference værdi 0, der ligger i RAM'en i en konstant benævnt "Ref". Ved denne referenceværdi er forstærkningen jf. ligning 3.2 afsnit 3.2 ca. lig 1. En tabel over de valgte variabelnavne, for de enkelte kanaler ses i tabel 7.1. Endelig sættes offset- og tærskelværdierne til en standard opstartsværdi på henholdsvis 7Fh og 30. Dette gør, at offset ved opstart har værdien 2,5V, og støj ikke overskrider tærskelværdierne.

Kanal	Variabelnavn
1	digmod1
2	digmod2
3	digmod3
4	digmod4

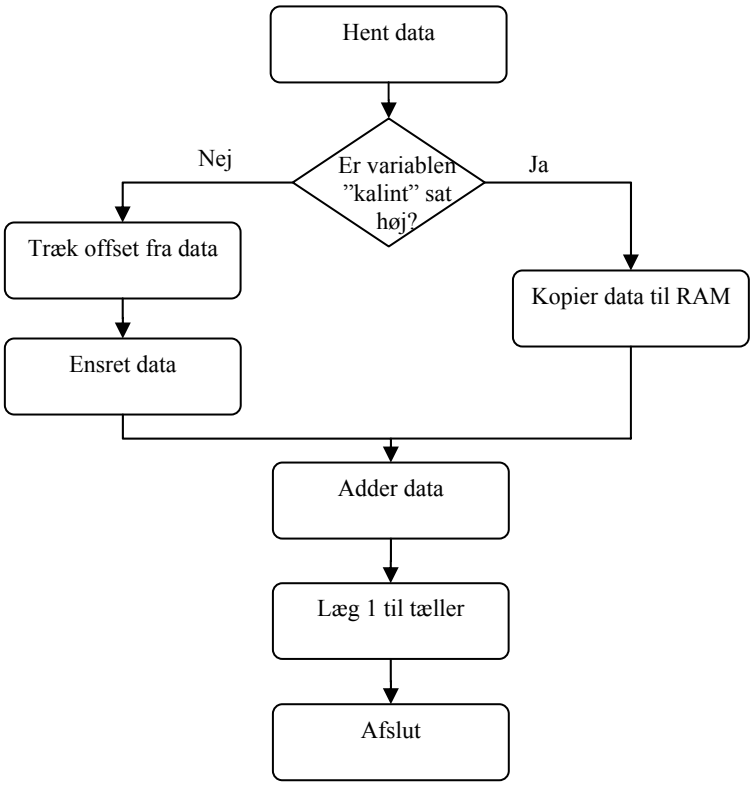
Tabel 7.1: De digitale modstandes variabelnavne

Afslutningsvis aktiveres rutinen bag interrupt 4, ACIA'ens interrupt, således mikrodatamaten er klar til at modtage yderligere kommandoer fra PC'en gennem ACIA'en.

7.2.2 Interruptrutine 1 og 2

Formålet med rutinerne bag interrupt 1 og 2, A/D-konverterens interrupts, er at hente data fra A/D-konverteren, behandle disse og lagre dem i mikrodatamatens interne dataregistre, når data er klar på A/D-konverteren. Endvidere skal interruptrutine 1 og 2, i det tilfælde at offset- og tærskelværdierne ønskes bestemt, kunne lagre dataene i mikrodatamatens interne dataregistre og samtidig lagre disse i

mikrodatamatens RAM. Flowdiagrammet for interrupt rutine 1 og 2 er identiske og er illustreret i figur 7.8.

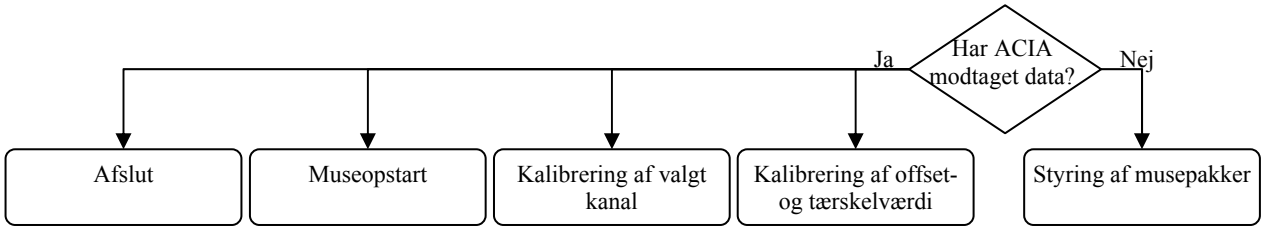


Figur 7.8: Flowdiagram for interrupt rutine 1 og 2

Når A/D-konverteren har data for kanal 1 og 2 eller kanal 3 og 4 klar på sin bufferkreds, aktiveres henholdsvis interrupt 1 eller 2, hvorved interruptrutinen henter dataene på bufferkredsen og placerer dem skiftevis i dataregister D4. Da interruptrutine 1 og 2 benyttes både ved indsamling af data til afsendelse af musepakker, ved kalibrering af kanaler og ved bestemmelse af offset- og tærskelværdier, er det fundet nødvendigt at dele rutinerne op i to dele.
Til denne opdeling benyttes den globale variabel ”kalint”. Denne sættes høj i det tilfælde at proceduren ”kalibrering af offset- og tærskelværdier” er opstartet og lav i de to øvrige tilfælde.
Er ”kalint” således høj, vil dataene, der nu ligger i dataregister D4, blive kopiereret over i mikrodatamatens RAM, således at disse data er tilgængelige på et senere tidspunkt. Herefter adderes dataene for de enkelte kanaler sammen med eventuelle forrige data, placeret i dataregister D0 til D3, og placeres igen i dataregistrene D0 til D3, der svarer til henholdsvis kanal 1 til 4.
For at kunne holde styr på, hvor mange værdier der er adderet sammen i dataregistrene, benyttes en tæller. Tællerværdien skal være en global variabel benævnt henholdsvis ”taelint1” og ”taelint2”, idet andre dele af assemblerkoden skal have adgang til denne værdi. Efter tællerværdien er opdateret, afsluttes interruptrutinen.
Er den globale variabel ”kalint” derimod lav, skal offsetværdierne, der er bestemt i proceduren ”Kalibrering af offset- og tærskelværdier”, trækkes fra de data der ligger i dataregister D4. Herefter ensrettes dataene for de enkelte kanaler, hvorefter disse adderes sammen med eventuelle forrige data, placeret i dataregister D0 til D3, og placeres igen i dataregistrene D0 til D3.
For at kunne holde styr på, hvor mange værdier der er adderet sammen i dataregistrene, benyttes igen tællerne ”taelint1” og ”taelint2”, idet andre dele af assemblerkoden stadig skal have adgang til denne værdi. Efter tællerværdien er opdateret, afsluttes interruptrutinen.

7.2.3 Interruptrutine 4

ACIA'en kan interrupte ved både afsendt og modtaget data. Når CPU'en modtager dette interrupt aktiveres interruptrutine 4. Denne rutine er illustreret i figur 7.9.



Figur 7.9: Flowdiagram over interruptrutine 4

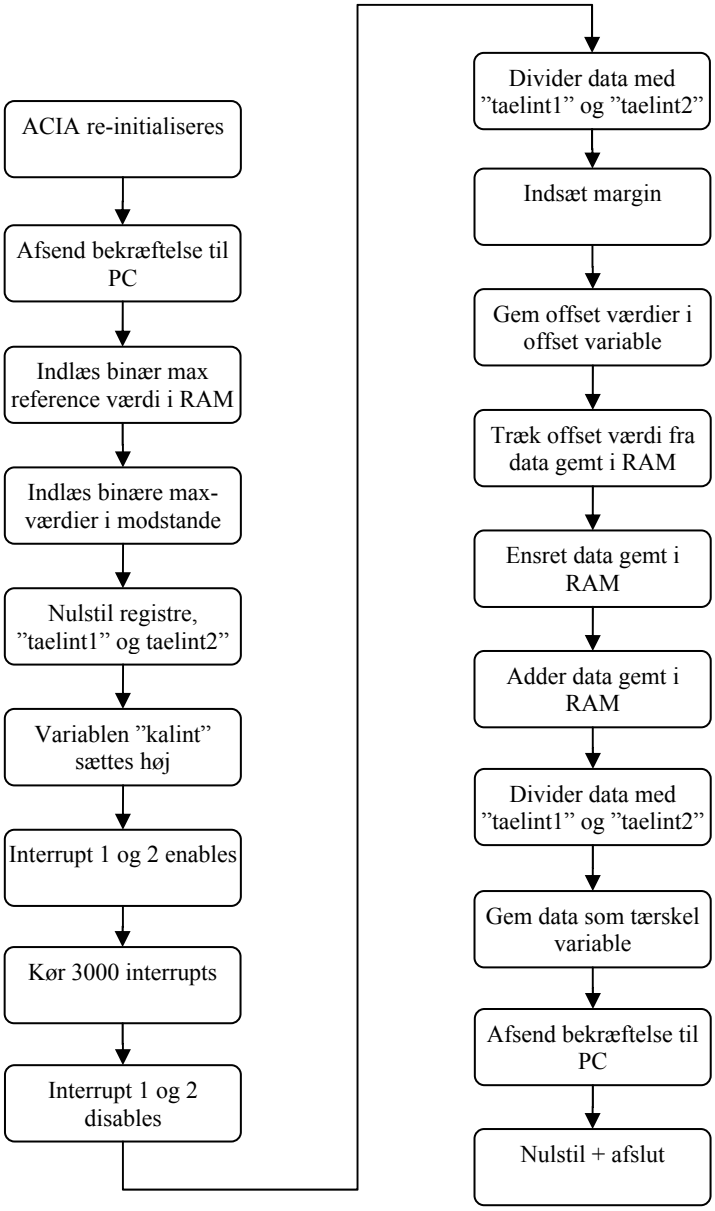
Interruptrutinen starter med at undersøge, hvorvidt ACIA'en har modtaget data fra PC'en eller afsendt data til PC'en. Har ACIA'en afsendt data skal proceduren "Styring af musepakker" aktiveres.
Har ACIA'en derimod modtaget data, vil disse data bestå af et af seks specificerede tegn. Er dataene ikke et af disse seks tegn vil interruptrutinen afslutte.
Et af de seks tegn vil aktivere proceduren "Museopstart", et andet vil aktivere proceduren "Kalibrering af offset- og tærskelværdier", og de andre fire vil afgøre, hvilken af de fire kanaler der skal kalibreres under proceduren "Kalibrering af valgt kanal". En oversigt over de seks tegn ses i tabel 7.2.

Tegn	Betydning
"1"	"1" gemmes i et dataregister og "Kalibrering af valgt kanal" aktiveres
"2"	"2" gemmes i et dataregister og "Kalibrering af valgt kanal" aktiveres
"3"	"3" gemmes i et dataregister og "Kalibrering af valgt kanal" aktiveres
"4"	"4" gemmes i et dataregister og "Kalibrering af valgt kanal" aktiveres
"5"	Proceduren "Kalibrering af offset- og tærskelværdier" aktiveres
"6"	Proceduren "Museopstart" aktiveres

Tabel 7.2: De seks tegn med tilhørende betydning

7.2.4 Kalibrering af offset- og tærskelværdier

Formålet med proceduren "Kalibrering af offset- og tærskelværdier" er først at bestemme offsetværdierne for de fire kanaler, således en korrekt behandling af EMG-signalerne er mulig. Endvidere skal de fire kanalers tærskelværdier bestemmes således, at utilsigtede markørbevægelser og klik, ved uspændte muskler, undgås. Disse tærskelværdier skal bestemmes når brugeren slapper af i sine muskler således, at det støj der når ind til mikrodatamaten i dette tilfælde sættes lig tærskelværdierne. Det er derfor også nødvendigt, at denne måling foretages ved den højest mulige forstærkning således, at tærskelværdien sættes ud fra den maksimalt mulige støj. De digitale modstande skal således under målingen have værdien "refmax", der sættes til den binære værdi af 255.
Flowdiagrammet over proceduren "Kalibrering af offset- og tærskelværdier" er illustreret i figur 7.10.



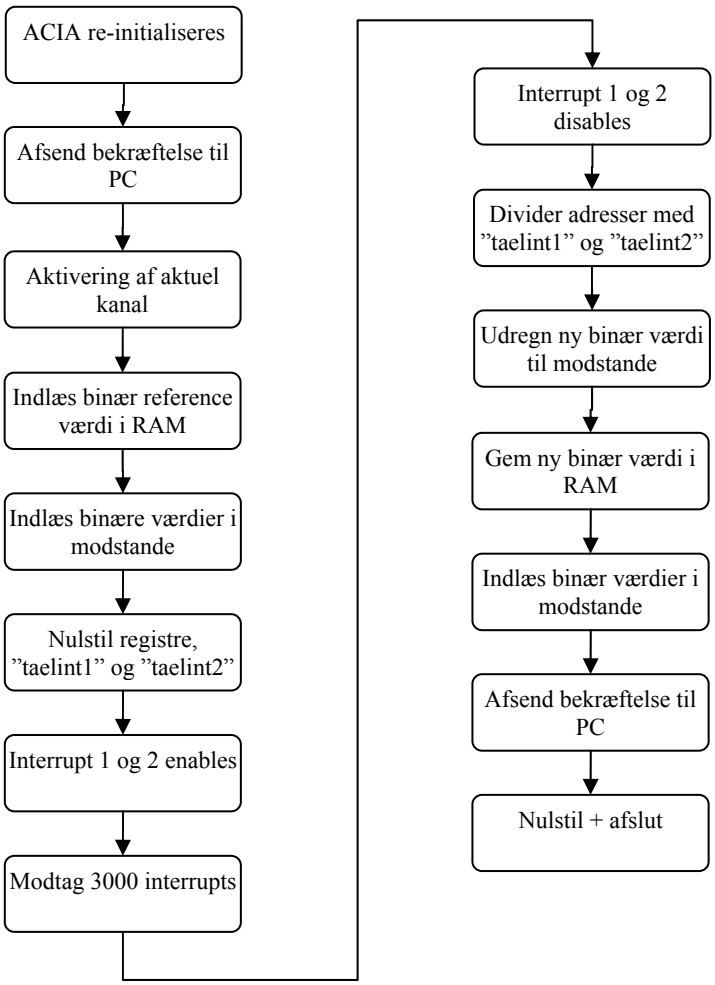
Figur 7.10: Flowdiagram over proceduren ”Kalibrering af offset og tærskelværdi”

Efter interrupt rutine 4 har aktiveret proceduren ”Kalibrering af offset- og tærskelværdier” re-initialiseres ACIA’en til kun at interrupte ved modtaget data. Dette gøres for at tage højde for det specielle tilfælde, at det vælges at kalibrere efter proceduren ”Museopstart” er aktiveret, hvorved mikrodatamaten ellers ville fortsætte med at afsende musepakker. Herefter afsendes en bekræftelse til PC’en i kraft af tegnet ”7”.
Da det er nødvendigt, at kalibreringen foretages med den binære referenceværdi ”refmax” indlæst i de digitale modstande, indlæses ”refmax” i de fire variabelnavne, jf. tabel 7.1. Værdierne beliggende i de fire variable, jf. tabel 7.2, indlæses nu serielt ind i de digitale modstande.
Herefter nulstilles de globale tællervariable ”taelint1” og ”taelint2” jf. afsnit 7.2.2 og dataregister D0 til D3 således, at kalibreringen er klar til at foretage nye målinger.

Den globale variabel ”kalint” sættes høj således, at interruptrutine 1 og 2 benyttes korrekt.
Interrupt 1 og 2 enables og sættes til at køre 3000 interrupts hver, før kalibreringsprocessen igen overtager. Da hver af de to interruptrutiner kører med en frekvens på 1000Hz, vil dette svare til et tidsrum på 3sek. Kalibreringsprocessen tester på ”taelint1”, og når denne opnår værdien 3000, disables interrupt rutine 1 og 2, hvorefter proceduren ”Kalibrering af offset- og tærskelværdier” genoptages. I dataregistrene D0 til D3 ligger nu summen af 3000 samplinger.
Dataene i dataregistrene divideres nu med værdien af enten ”taelint1” eller ”taelint2”, hvorved gennemsnitsværdierne og dermed også offsetværdierne er fundet.
Disse fire offsetværdier gemmes i de fire globale variable ”offset1” til ”offset4”.
Interrupt rutine 1 og 2 har, som beskrevet i afsnit 7.2.2, gemt hver eneste sampling i RAM’en. Fra samtlige samplinger trækkes nu offsetværdierne svarende til den enkelte kanal. Efterfølgende ensrettes samtlige samplinger. Herefter adderes alle samplinger kanalvis og lagres i dataregistrene D0 til D3.
Værdierne i D0 og D1 divideres med værdien af ”taelint1” og værdierne i D2 og D3 divideres med værdien af ”taelint2” således, at tærskelværdierne for samtlige kanaler er fundet. Til disse tærskelværdier lægges en margin, inden de gemmes i de fire globale variable ”taersk1” til ”taersk4”. Denne margin gør, at støjniveauet ikke overskrider de fastsatte tærskelværdier.
En bekræftelse på at offset- og tærskelværdierne nu er bestemt sendes til PC’en i form af tegnet ”7”.
Endeligt nulstilles de globale tællervariable ”taelint1” og ”taelint2”, dataregister D0 til D3 og variablen ”kalint” sættes lav, hvorefter proceduren afslutter.

7.2.5 Kalibrering af valgt kanal

Proceduren ”kalibrering af valgt kanal” har naturligvis samme formål som PC software kalibrerings proceduren beskrevet i afsnit 7.1.3. Procedurens flowdiagram er illustreret i figur 7.11.



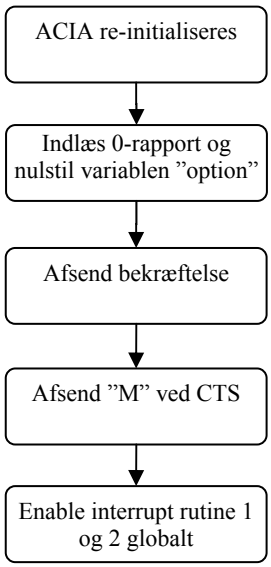
Figur 7.11: Flowdiagram over proceduren ”Kalibrering af valgt kanal”

Efter interruptrutine 4 har aktiveret proceduren ”Kalibrering af valgt kanal” reinitialiseres ACIA’en til kun at interrupte ved modtaget data. Dette gøres for at tage højde for det specielle tilfælde, at det vælges at kalibrere efter proceduren ”Museopstart” er aktiveret, hvorved mikrodatamaten ellers ville fortsætte med at afsende musepakker. Herefter afsendes en bekræftelse til PC’en i kraft af tegnet ”7”.
På baggrund af hvilket af de fire tegn der ligger i et af dataregistrene, jf. tabel 7.2, sættes proceduren ”Kalibrering af valgt kanal” til at kalibrere den korrekte kanal.
Da det er nødvendigt, at kalibreringen foretages med den binære referenceværdi ”ref” indlæst i den aktuelle digitale modstand, benyttes de fire variabelnavne, jf. tabel 7.1, til at holde styr på, hvilken værdi der er indlæst i de digitale modstande. Da den binære værdi for en enkelt kanal indlæses serielt i de digitale modstande sammen med de binære værdier for de tre øvrige kanaler, indlæses alle værdierne beliggende i de fire variable, jf. tabel 7.2, serielt ind i de digitale modstande. Når en enkelt kanal således skal kalibreres, lægges den binære referenceværdi på den aktuelle kanals variabel, hvorved de øvrige værdier i de digitale modstande ikke ændres.
Herefter nulstilles de globale tællervariable ”taelint1” og ”taelint2” jf. afsnit 7.2.2 og dataregister D0 til D3 således, at kalibreringen er klar til at foretage nye målinger.
Interrupt 1 og 2 enables og sættes til at køre 3000 interrupts hver, før kalibreringsprocessen igen overtager, og interrupt 1 og 2 disables igen.

I dataregistrene D0 til D3 ligger nu summen af 3000 behandlede samplinger. Ved disse 3000 samplinger har brugeren spændt sine muskler til et niveau, som han/hun ønsker skal være det maksimale niveau.
Resultatet i det aktuelle dataregister divideres med værdien af enten ”taelint1” eller ”taelint2”, hvorved en gennemsnitsværdi er fundet. På baggrund af denne værdi udregnes nu en ny binær værdi til den aktuelle digitale modstand.
Den nye binære værdi gemmes i den korrekte variabel, hvorefter binær værdierne for de fire variable på ny læses ud i de digitale modstande.
En bekræftelse på at kanalen nu er kalibreret afsendes til PC’en i form af tegnet ”7”. Endeligt nulstilles de globale tællervariable ”taelint1” og ”taelint2” samt dataregister D0 til D3, hvorefter proceduren afslutter.

7.2.6 Museopstart

Formålet med proceduren ”Museopstart” er, at informere PC’en om, at det er en mus, der er koblet til COM1 porten. Samtidig vil afsendelse af denne information starte afsendelse af musepakker. Procedurens flowdiagram er illustreret i figur 7.12.



Figur 7.12: Flowdiagram over proceduren ”Museopstart”

Når proceduren ”Museopstart” aktiveres fra PC’en, vil proceduren som det første reinitialisere ACIA’en, således denne interrupter både ved modtaget og afsendt data.
Efterfølgende indlæses tre musepakker, svarende til en 0-rapport, på variablerne i tabel 7.3, og den globale variabel ”option” nulstilles.

Musepakke nummer	Variabelnavn
1	muspak1
2	muspak2
3	muspak3

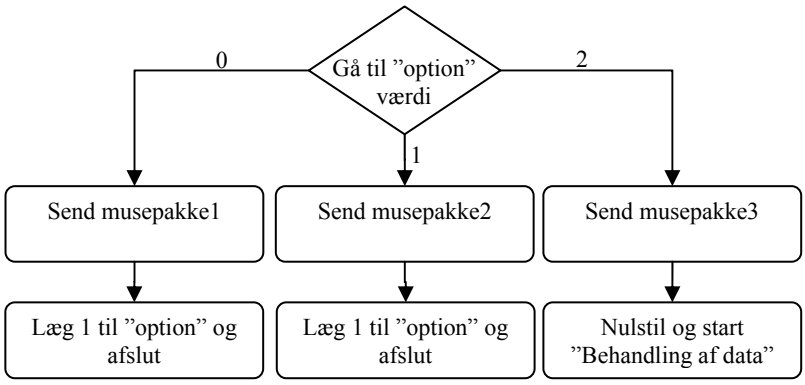
Tabel 7.3: Musepakkernes variabelnavne

Dette gøres for at tage højde for det tilfælde, at proceduren ”museopstart” har været kørt tidligere. Nødvendigheden af dette, og den globale variabel ”option”, uddybes i afsnit 7.2.7.

Herefter afsendes en bekræftelse til PC'en, på at "Museopstart" er aktiveret. Proceduren vil nu afvente, at PC'en forespørger tegnet "M". Ved denne forespørgsel sættes CTS på ACIA'en lav, hvorved proceduren afsender tegnet "M".
Endelig enables interruptrutine 1 og 2 globalt, således at en af disse vil aktiveres hver gang A/D-konverterens kontrollogik afsender et interrupt.

7.2.7 Styring af musepakker

Formålet med denne procedure er at styre afsendelsen af musepakkerne, således dette sker i den rigtige rækkefølge. Endvidere sørger proceduren for at aktivere proceduren "Behandling af data" på det rette tidspunkt. Proceduren "Styring af musepakker"s flowdiagram er illustreret i figur 7.13.

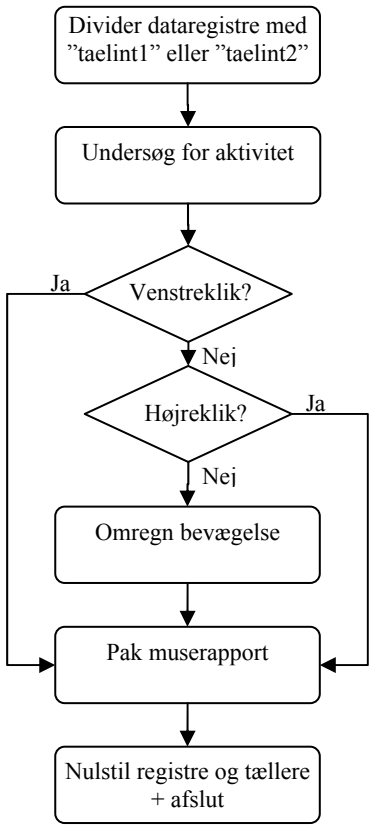


Figur 7.13: Flowdiagram over proceduren "Styring af musepakker"

Proceduren "Styring af musepakker" aktiveres via interrupt rutine 4, når ACIA'en har afsendt data. Når proceduren aktiveres aflæses værdien af variablen "option". Første gang proceduren aktiveres har "option" værdien 0, jf. afsnit 7.2.6, hvorfor musepakke 1 afsendes. Herefter lægges 1 til "option", og proceduren afsluttes.
Afsendelsen af første musepakke forårsager en ny interrupt fra ACIA'en, der igen aktiverer proceduren "Styring af musepakker". Denne gang har "option" værdien 1, hvorfor anden musepakke afsendes til ACIA'en. Herefter lægges igen 1 til "option", og proceduren afsluttes.
Afsendelsen af anden musepakke forårsager igen et interrupt fra ACIA'en, der på ny aktiverer proceduren "Styring af musepakker". Denne gang har "option" værdien 2, hvorfor tredje musepakke afsendes til ACIA'en. Herefter aktiveres proceduren "Behandling af data", og dataregistre og "option" nulstilles.
Denne proces vil køre i ring, og hele tiden sende nye musepakker til ACIA'en, idet afsendelsen af én musepakke automatisk vil aktivere afsendelsen af endnu en musepakke. Proceduren "Behandling af data" sørger for at disse musepakker hele tiden er klar til afsendelse.
Startes proceduren "Museopstart" efter denne har været kørt før, er det nødvendigt, at "option" bliver nulstillet, og en 0-pakke bliver placeret på variablerne jf. tabel 7.3. Dette bliver foretaget af proceduren "Museopstart".

7.2.8 Behandling af data

Proceduren "Behandling af data" bliver aktiveret af mulighed 2 i proceduren "Styring af musepakker", og har til opgave at behandle de data der frembringes af interrupt 1 og 2, således disse omdannes til musepakker. Proceduren "Behandling af data"s flowdiagram er illustreret i figur 7.14.



Figur 7.14: Flowdiagram over proceduren ”Behanding af data”

Som det første bestemmes gennemsnitsværdien for alle fire kanaler. Dette gøres ved at dividere værdierne i dataregister D0 og D1 med værdien af ”taelint1”, og dividere værdierne i dataregister D2 og D3 med værdien af ”taelint2”. Herefter undersøges, hvorvidt gennemsnitsværdien for de enkelte kanaler er over eller under tærskelværdierne. Er en værdi under den tilhørende tærskelværdi sættes denne kanals værdi til 0, og er en værdi over tærskelværdien ændres intet ved denne kanals værdi.
Efterfølgende undersøges, hvorvidt de fire kanalers værdier forårsager klik eller bevægelse. Uanset om der er tale om klik eller bevægelse, omregnes disse data til data relevant i forhold til musepakker, hvorefter den færdige muserapport pakkes og musepakkerne lægges på adresserne angivet i tabel 7.3.
Endeligt nulstilles dataregistrene D0 til D3 og ”taelint1” og ”taelint2” nulstilles, hvorefter proceduren afslutter.

7.3 Delkonklusion

PC-programmet blev testet ved at køre det på en computer, der var tilsluttet en anden computer vha. et serielkabel. På den anden computer kørte et terminalprogram, der kunne sende til og modtage data fra computeren med PC-programmet. Hvis det i PC-programmet vælges at uploade programmet til mikrodatamaten, kan det i terminalprogrammet ses, om den ønskede data når frem, hvilket viste sig at være tilfældet i den pågældende test. Endvidere kan det ved opstart af museprogrammet konstateres, at der fra PC-programmet bliver afsendt et ”6” til terminalprogrammet, og at PC-programmet venter på at modtage et ”7”. Sker dette ikke melder PC-programmet fejl. Ved kalibrering kan det i terminalprogrammet ses, at der fra PC-programmet

bliver afsendt tegnet for den pågældende kanal eller tegnet ”5” for indstilling af tærskelværdien. Efterfølgende venter PC-programmet på to ”7”’er for henholdsvis påbegyndt og afsluttet kalibrering. Ved timeout melder PC-programmet fejl. PC-programmet fungerer således efter hensigten skitseret i figur 7.1.
I den endelige test af PC-programmet blev mikrodatamaten tilsluttet i stedet for computeren med terminalprogrammet. Det kunne ligeledes her konstateres, at mikrodatamaten, programmet til mikrodatamaten og PC-programmet kommunikerer efter hensigten, hvilket kan ses i et videoklip på den vedlagte CD [CD\:Test\videoklip.(avi/mpg)]. Videoklippet viser at musemarkøren bevæger sig 100 pixels i hver retning, hvilket svarer til 100 muserapporter, idet hver rapport bevæger markøren én pixel. Endvidere viser videoklippet, at der afsendes 40 muserapporter i sekundet, idet en bevægelse på 100 pixels tager 2,5sek. Kildekoden for PC-programmet fremgår af appendiks H.
Test af programmet til mikrodatamaten har endvidere vist, at det er muligt at bestemme offset- og tærskelværdier efter hensigten jf. appendiks I.2 og I.3.
Test af proceduren ”kalibrering af valgt kanal” er endvidere blevet foretaget under debugging af programmet. Under debugging viste det sig, at de korrekte data blev lagt i RAM’en, og det blev med oscilloscop påvist, at der blev sendt data til latch-kredsende DM74LS373 placeret ummiddelbart før de digitale modstande DS1267. De digitale modstande blev dog ikke korrekt indstillet, som nævnt i afsnit 6.6. Det kan af dette konkluderes, at proceduren ”kalibrering af valgt kanal” tilsyneladende fungerer korrekt, men systemet lider af en hardwarefejl i opsætningen af de digitale modstande. Det har dog af tidsmæssige årsager ikke været muligt at undersøge hvorvidt de data, der blev sendt til latch-kredsene, er korrekte, og det kan således ikke fuldstændigt udelukkes, at problemet er en softwarefejl.
Kildekoden for programmet til mikrodatamaten fremgår af appendiks I.
Det er valgt i kildekoden, til mikrodatamatens programmel, at udkommentere proceduren ”kalibrering af valgt kanal”, da denne ikke implementeres i det endelige programmel.

8 Konklusion

Formålet med dette projekt var at konstruere et system, der via EMG-signaler gør brugeren i stand til styre en musemarkør.
Den EMG-styrede mus er primært rettet mod fysisk handikappede brugere, der ikke har mulighed for at anvende en almindelig mus.
Systemet kan inddeles i følgende hoveddele: Hardware, software til mikrodatamaten og PC-software.

Virkemåden for systemet er: Signaler introduceres på EMG-elektroderne. Disse signaler sendes derefter gennem en galvanisk adskillelse, hvorefter signalerne forstærkes og filtreres. Signalerne konverteres fra analog til digital, og mikrodatamatens software konverterer disse oplysninger til tilsvarende musepakker, der styrer musemarkøren på skærmen.

Kravene til systemet er:

- EMG-signalerne skal forstærkes til maksimalt $5V_{PP}$ gennem feedback fra mikrodatamaten.
- Der skal indsættes en galvanisk adskillelse mellem brugeren og resten af systemet af sikkerhedshensyn. Forstærkningen gennem adskillelsen må maksimalt have en ulinearitet på $\pm 3dB$.
- Høj- og lavfrekvent støj skal dæmpes gennem et filter, der samtidig skal virke som anti-aliasing filter.
- De analoge signaler skal konverteres til et digitalt signal.
- Mikrodatamatens software skal kunne konvertere dette signal til musepakker.
- Mikrodatamatens softwaren skal kunne kalibrere hver kanal.
- PC-softwaren skal kunne uploade et program til mikrodatamaten.

Systemet, der er blevet konstrueret til dette projekt kan opfylde følgende krav:

- EMG-signalerne bliver forstærket til $5V_{PP}$, ikke gennem feedback fra mikrodatamaten, men manuelt vha. analoge potentiometre.
- Der er indsat en galvanisk adskillelse mellem brugeren og resten af systemet. Forstærkningens ulinearitet ligger under $\pm 3dB$.
- Høj- og lavfrekvent støj bliver gennem et anti-aliasing filter dæmpet.
- De analoge signaler bliver konverteret til et digitalt signal.
- Mikrodatamatens software konverterer det digitale signal til musepakker.
- PC-softwaren kan uploade et program til mikrodatamaten.

Gennem tests er det blevet bekræftet, at EMG-signalerne kan forstærkes op til $5V_{PP}$.
Kravet om at forstærkningen af signalerne skal kunne indstilles gennem feedback fra mikrodatamaten er ikke opfyldt. Løsningen med at implementere en digitalt indstillelig modstand til at regulere forstærkningen af EMG-signalet blev dog udført, men tests har vist, at denne løsning desværre ikke fungerede korrekt. Årsager til at de digitale modstande ikke fungerer korrekt kan være fejl i assemblerkoden, således at de værdier der indlæses til de digitale modstande, ikke er korrekte. Selve hardwaredelen af de digitale modstande kan være opsat ukorrekt, således de digitale

modstande ikke indstilles selvom de modtager det korrekte signal fra mikrodatamaten. Endelig kan det være timingen til kredsen, der ikke lever op til skiftet på databussen.
Af tidsmæssige årsager blev fejlen i ved kalibreringen af de fire kanaler ikke fundet. I stedet er det valgt at indsætte potentiometre, hvor de digitale modstande skulle være, og disse skal således indstilles manuelt i forhold til den ønskede signalstyrke.
Det er efter hardwaretidsfristen i dette projekt erfaret, at der er bedre alternativer til regulering af forstærkningen digitalt, end den i dette projekt valgte. Men grundet tidsfristen er det valgt ikke at behandle disse mulige løsninger.
Den indsatte galvaniske adskillelse med forstærkning har gennem tests vist sig at fungere korrekt. Disse tests viser også at forstærkningen er indenfor linearitetens grænser.
Filteret er blevet testet, og det er konstateret, at filteret lever op til de opstillede krav. Frekvenser over 500Hz bliver dæmpet med over 40dB. Filteret virker således også som et anti-aliasing filter, da den efterfølgende A/D-konvertering sampler med 1000Hz.

A/D-konverteringen skulle oprindeligt have været uden offset og dæmpning af signalet. Gennem tidlige tests blev det dog erfaret, at dette ikke kunne lade sig gøre med den fra starten valgte A/D-konverter. Det er af denne grund, at både filteret og den galvaniske adskillelse med forstærkning, er designet til at kunne behandle et signal på op til $10V_{PP}$. Faktum er, at det blev erfaret for sent i processen, at A/D-konverteren ikke levede op til dette krav. Af denne grund er indgangen på A/D-konverteren blevet modificeret til at konvertere signalet, således dette svinger mellem 0 og $5\hat{V}$, i stedet for $\pm 5\hat{V}$. Havde tiden tilladt det, havde det dog været mere hensigtsmæssigt at vælge en ny A/D-konverter. Denne nye A/D-konverter skulle således have været i stand til at konvertere signaler på $\pm 5\hat{V}$.
Tests af A/D-konverteringen har vist, at de analoge signaler omsættes til digitale værdier, med en vis støj. Denne støj, som tests viser A/D-konverteren selv er kilde til, bevirker at signalerne ikke kan aflæses korrekt af A/D-konverteringen. Det er dog ikke gengivelsen af signalets karakter, der er betydende i forhold til oplysningerne softwaren i mikrodatamaten skal benytte. Her er det signalets amplitude, der er den afgørende faktor, hvor støj har en mindre betydning. Derfor konstateres det, at A/D-konverteringen lever op til de stillede krav.

Softwaredelen af den EMG-styrede mus er blevet testet og er fundet tilfredsstillende. Det er således muligt for PC-softwaren at uploade mikrodatamatprogrammet til mikrodatamaten samt kommunikere med dette program.
Programmet til mikrodatamaten er i stand til at afsende musepakker på baggrund af konverterede EMG-signaler, samt bestemme offset- og tærskelværdier. Desværre har det ikke være muligt at implementere den del af assemblerkoden, der styrer kalibreringen af de enkelte kanaler. Dette skyldes som sagt en fejl i enten hardwaren eller programmet til mikrodatamaten.

Den endelige test af den EMG-styrede mus har vist at brugeren kan styre musemarkøren samt emulere klik, som beskrevet i kravsspecifikationen, vha. EMG-signaler. Det kan dog være nødvendigt for brugeren at øve sig en anelse, samt justere forstærkningen vha. potentiometrene for at opnå maksimal kontrol over musemarkøren.
Hvorledes brugeren skal installere og benytte den EMG-styrede mus er beskrevet i manualen placeret bagerst i rapporten.

Den endelige konklusion på projektet må blive, at dette er forholdsvis vellykket. Vellykket, idet det er lykkedes at konstruere en EMG-styret mus, men dog kun forholdsvis vellykket, idet systemet ikke kan kalibreres som opstillet i kravsspecifikationen.

9 Litteraturliste

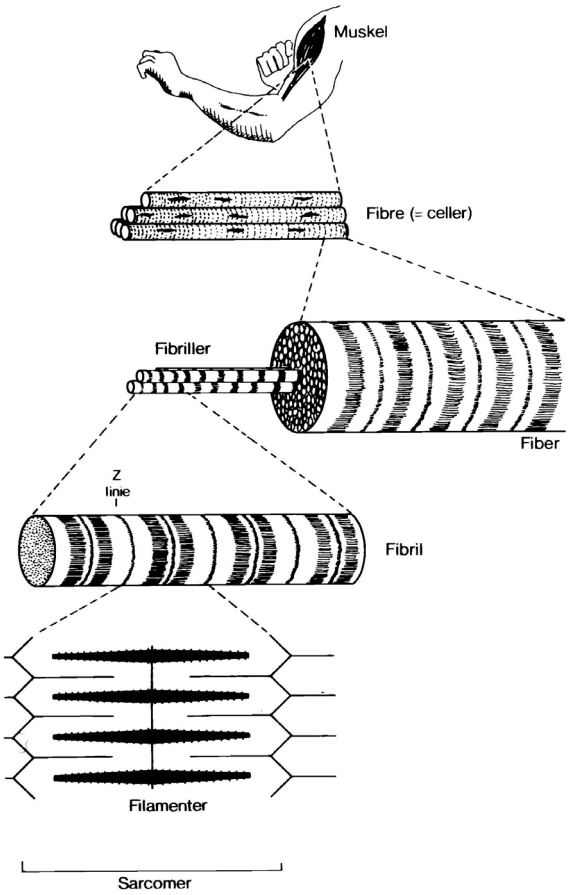
Reference	Sedra	Fys
Titel	Microelectronic Circuits	Fysiologi
ISBN	0-19-511690-9	87-87661-527
Forlag	Oxford University Press	Nucleus
Udgave	4.	2. udgave 3. oplag
Årstal/Dato	1998	1984
Forfatter	Sedra, Adel S., Smith, K.	Erling Asmussen mfl.
Link	www.oup.com	-
Reference	**Filter**	**Zone**
Titel	Analog filter design	
ISBN	4-8337-0091-3	
Forlag	Holt-Saunders	
Udgave	-	
Årstal/Dato	1982	4/4-2002
Forfatter	M. E. Van Valkenburg	
Link	-	zone.ni.com
Reference	**Bio**	**SEMG**
Titel	-	-
ISBN	-	-
Forlag	-	-
Udgave	-	-
Årstal/Dato	14/3-2002	14/3-2002
Forfatter	-	-
Link	www.delsys.com/library/papers/biomachanics.pdf	www.delsys.com/library/papers/SEMGintro.pdf
Reference	**ADC**	**MICRO**
Titel	High-speed analog-to-digital conversion	Microprocessors systems design
ISBN	0-12-209048-9	0-534-94822-7
Forlag	Academic press, Inc.	PWS Publishing company
Udgave	1.	3.
Årstal/Dato	1991	1997
Forfatter	Michael J. Demler	Alan Clements

A Elektromyografi

Elektromyografi (EMG) er et udtryk for måling af den elektriske muskelaktivitet under en muskelkontraktion. Til forståelse af hvorledes denne muskelkontraktion foregår, gives en kort introduktion til skeletmusklers opbygning og virkemåde.

A.1 Muskel

Skeletmusklen, den eneste af i alt tre muskeltyper der er underlagt viljens kontrol, er opbygget af op til adskillige hundredetusinde muskelfibre. En muskelfiber er opbygget af myofibriller, der igen indeholder myofilamenter, som er delt op i en række ens områder kaldet sarkomerer (se figur A.1).

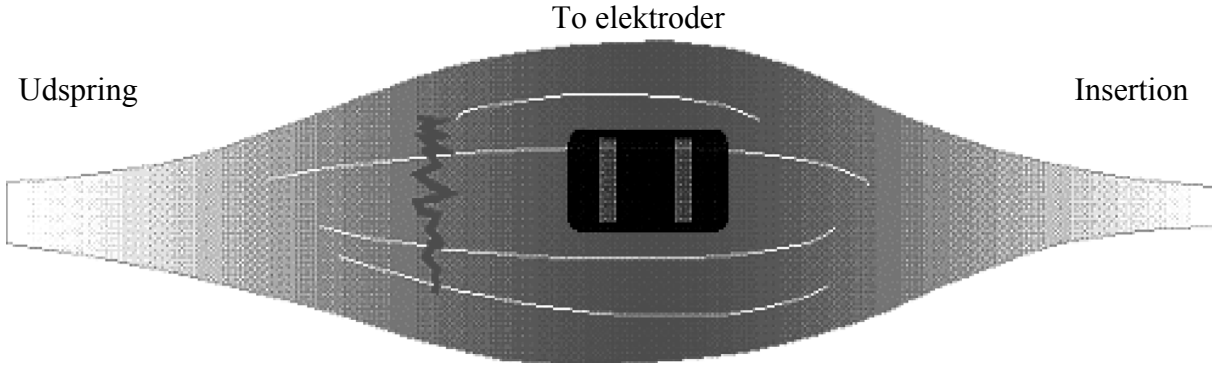


Figur A.1: Opbygning af skeletmuskel [Fys. S.100]

Et sarkomer består af tykke og tynde myofilamenter, og er afgrænset af to Z-skiver bestående af tynde myofilamenter. Ved en muskelkontraktion mindskes afstanden mellem Z-skiverne, hvilket sker ved en elektrisk aktivering af en muskelfiber ved at en nerves elektriske impuls overføres kemisk til muskelfiberen.
Denne elektriske aktivering af muskelfiberen finder sted midt på muskelfiberen ved et område kaldet den motoriske endeplade [Fys s. 99-100].

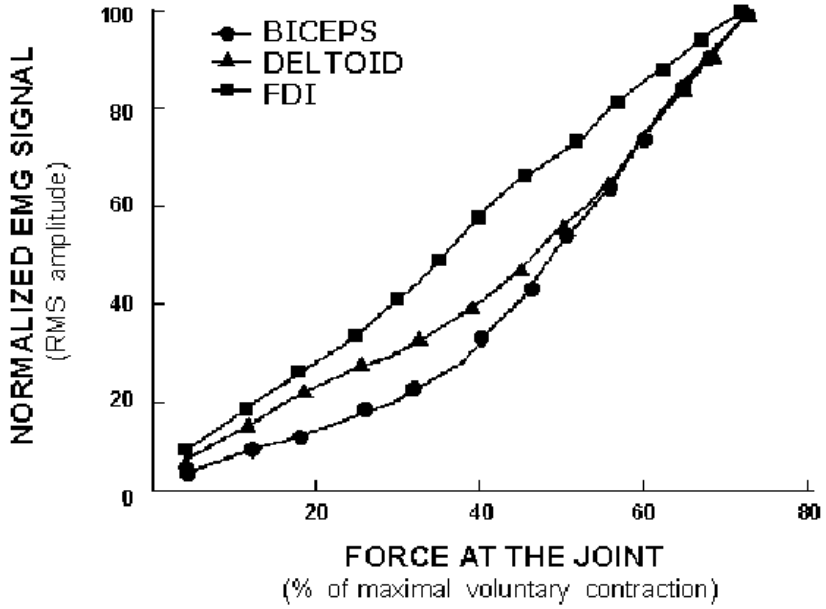
A.2 Signalet

Den elektriske impuls, der via nerven overføres til musklen, kan måles vha. en overflade-elektrode der sættes på en af brugerens muskler. Dette kan også ske ved hjælp af en wire-elektrode, hvilket giver et kraftigere signal. Den sidste metode afvises dog, idet den kræver penetrering af huden og den underliggende muskel.
Placeringen af overflade-elektroden er afgørende, idet der findes forskellige områder på musklen, hvor det på baggrund af signal-styrken er uhensigtsmæssigt at placere elektroden. Da det kan være svært at lokalisere visse af disse områder benyttes en tommelfinger-regel der dikterer, at elektroden placeres midt mellem musklens insertion og udspring.
Foruden elektrodens placering er også orienteringen af elektroden vigtig. Orienteringen skal være vinkelret med de aktuelle muskelfibre, for at opnå kontakt med størst mulig antal fibre (se figur A.2). Endvidere er der nødvendigt med to elektroder placeret parallelt ved siden af hinanden. Grunden til dette forklares i slutningen af dette kapitel[SEMG s.7].



Figur A.2: Elektrodens korrekte orientering[SEMG s.7]

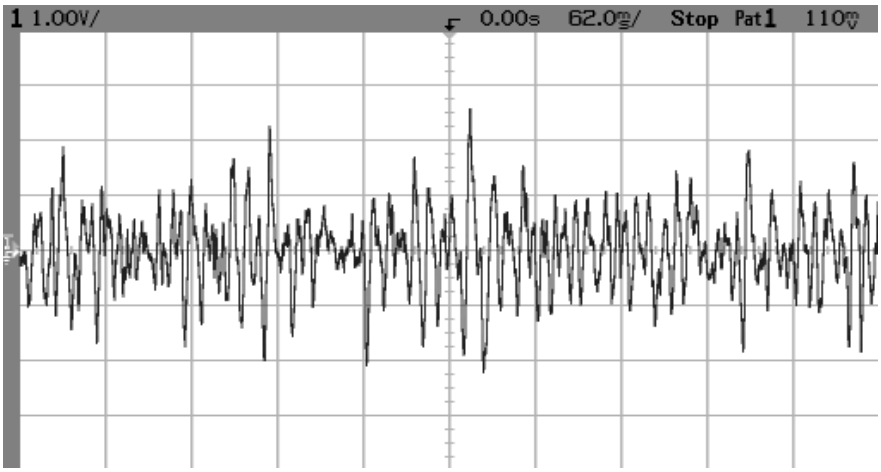
Til forståelse af sammenhængen mellem EMG-signalet og kraften benyttes figur A.3. Der viser en tilnærmelsesvis linearitet mellem de to størrelser ved de tre valgte muskler.



Figur A.3: Sammenhængen mellem EMG-signalet og den kraft hvormed en muskel spændes ved statisk arbejde for tre forskellige muskler [BIO s.14]

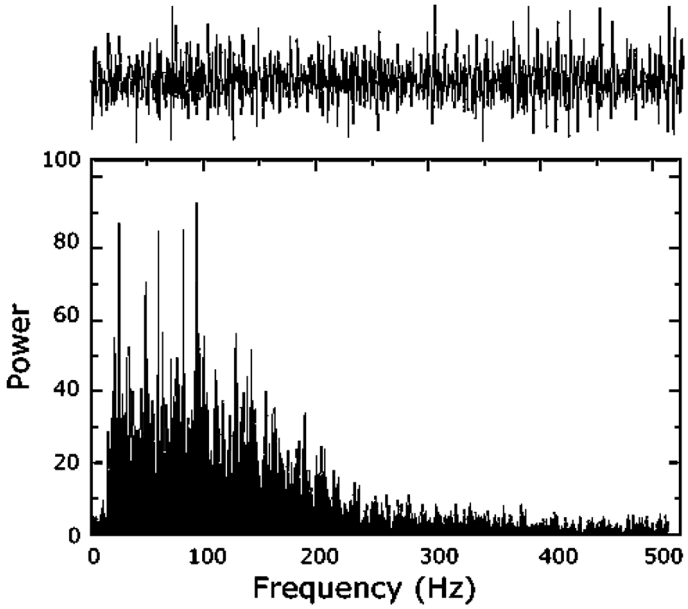
A.3 Signal/støjforhold

Amplituden af EMG-signalet ligger teoretisk i størrelsesordenen $0-5m\hat{V}$.
En praktisk måling af et EMG-signal ses i figur A.4. Af figuren ses at EMG-signalets maksimale amplitude, ved måling på biceps, er ca. $2,5m\hat{V}$.



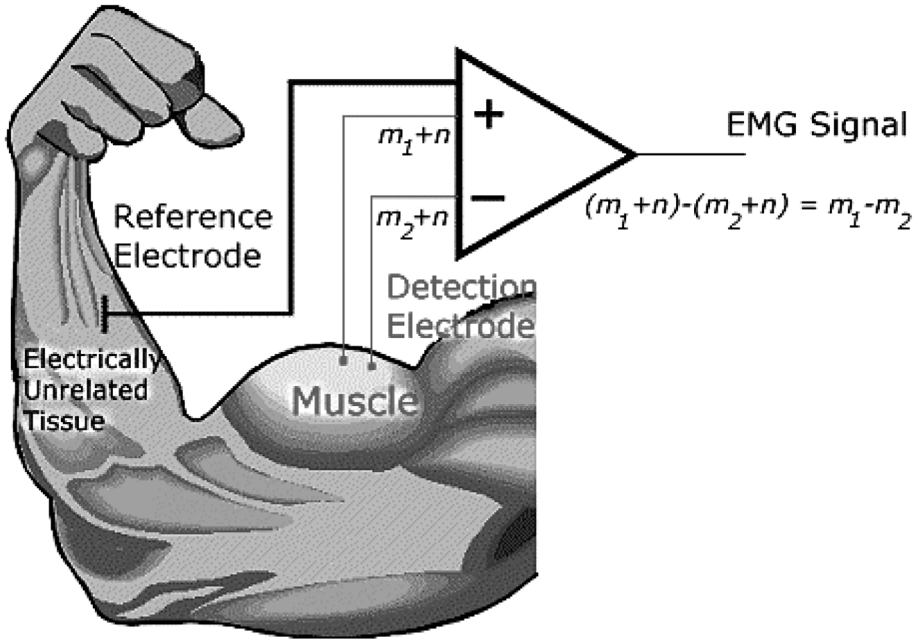
Figur A.4: EMG-signal fra biceps med en forstærkning på 1000

Størstedelen af energien i signalet ligger i området 0-500Hz, og er særligt koncentreret i området 50-150Hz, hvilket ses i figur A.5.



Figur A.5: EMG-signal samt dettes frekvensspektrum [SEMG s.2]

Da amplituden af EMG-signalet er lavt, opstår der problemer med støj fra omgivelserne. Denne støj vil hovedsageligt stamme fra elektromagnetisk stråling fra nærliggende elektriske apparater, samt den "bevægelsesstøj" der fremkommer ved kontaktfladen mellem huden og elektroden. Da støjen ligger i det samme frekvensområde som signalet, er det ikke praktisk muligt at filtrere det væk, idet også signalet ville blive filtreret fra.
En løsning af dette problem er implementeringen af en instrumenteringsforstærker (se figur A.6).



Figur A.6: En grafisk oversigt over en differentialforsærkers opsætning. EMG signalet er markeret med "m", mens støjsignalet er markeret "n"[SEMG s.4]

Instrumenteringsforstærkeren modtager to signaler, samt et reference-signal. De to signaler, med støj, trækkes fra hinanden hvorved støjen forsvinder. Herefter forstærkes forskellen af de to rene signaler.

B RS-232

I en almindelig PC tilsluttes musen enten gennem en serielport, en PS/2 port eller en USB-port. Som beskrevet i kravspecifikationen er det valgt at benytte en serielport, da denne eksisterer i stort set alle computere. En serielport anvender RS-232 kommunikationsstandarden, hvorfor det er valgt at beskrive denne inden beskrivelsen af selve musens virkemåde.

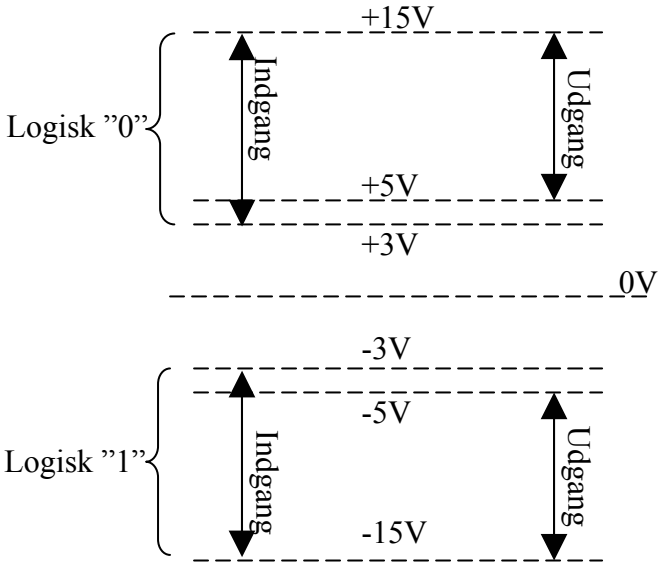
Mekanisk set består RS-232 af 25 ledere, men i praksis anvendes i langt de fleste tilfælde kun 9 ledere, hvilket også gælder for en mus. I tabel B.1 ses en oversigt over 9-bens RS-232.

Ben nr.	Navn	RS-232	Retning	Beskrivelse	Anvendte signaler i en mus
1	CD	CF	💻←🖱	Carrier Detect	-
2	RXD	BB	💻←🖱	Recieve Data	Data fra mus til PC
3	TXD	BA	💻→🖱	Transmit Data	+12V
4	DTR	CD	💻→🖱	Data Terminal Ready	-
5	GND	AV	💻↔🖱	Ground	0V
6	DSR	CC	💻←🖱	Data Set Ready	-
7	RTS	CA	💻→🖱	Request To Send	Klarsignal fra PC til mus, +12V
8	CTS	CB	💻←🖱	Clear To Send	-
9	RI	CE	💻←🖱	Ring Indicator	-

Tabel B.1: Musens anvendelse af RS-232

Som det ses anvendes blot 4 ud af de 9 mulige ben, hvoraf det ene kan undværes, da det udelukkende bruges til spændingsforsyning til musen.

RS-232 opererer indenfor nogle andre spændingsområder end almindelige TTL-signaler. Dette skyldes, at RS-232 skal have en længere rækkevidde og dermed har et behov for at være mere støjresistent. På figur B.1 ses spændingsniveauerne for henholdsvis ind- og udgang for RS-232 standarden.



Figur B.1: Spændingsniveauerne for henholdsvis ind- og udgang for RS-232 standarden

I praksis vil spændingerne ligge omkring ±12V. Bemærk at der er byttet rundt på potentialerne for logisk 0 og 1 i forhold til TTL.

B.1 Musens virkemåde

For at musen kan kommunikere med PC'en gøres brug af en driver. Denne driver er implementeret i næsten alle operativsystemer og er oftest baseret på MS Mouse Standard.
Når serielporten i PC'en er klar, vil CTS gå logisk høj. Dette medfører, at musen svarer ved at sende ASCII-værdien 77 svarende til ”M” gennem RXD til PC'en. Herved initialiseres musedriveren, hvorefter PC'en er klar til at modtage data fra musen.

Hvert sekund sender musen 40 rapporter til PC'en indeholdende information om ændringen i sin position. Hver af disse rapporter består af tre pakker jf. tabel B.2. Som det fremgår, begynder hver pakke med en startbit umiddelbart efterfulgt af bit D7, der skal være 1 i det tilfælde, at der benyttes 8 databit. Dernæst kommer D6, der har til formål at synkronisere begyndelsen af hver rapport. Efterfølgende kommer D5 og D4, der i musepakke 1, beskriver henholdsvis venstre og højre musetasts tilstand. Hvis den pågældende tast er aktiveret, vil bitværdien være 1.

	Start	D7	D6	D5	D4	D3	D2	D1	D0	Stop
1	0	X	1	LB	RB	Y7	Y6	X7	X6	1
2	0	X	0	X5	X4	X3	X2	X1	X0	1
3	0	X	0	Y5	Y4	Y3	Y2	Y1	Y0	1

Tabel B.2: Opbygning af en rapport jf. MS mouse standard

Ændringen i både X- og Y-retningen er beskrevet ved hver 8 bit, hvoraf det første angiver fortegn. Dette giver mulighed for decimalværdier på ±127. Hver pakke afsluttes af en, halvanden eller to stopbit, hvor det er valgt at benytte én. Angivelsen er i 2's komplement. [Mouse]

Overførselshastigheden fra mus til PC er 1200bps, hvilket som tidligere nævnt svarer til 40 rapporter, da hver rapport fylder 3 gange 10 bit.

Opsumering

Gennem kapitlet er følgende fakta fastlagt:

- Antal rapporter per sekund: 40
- Størrelse af rapport: 30 bit
- Antal databit: 8
- Antal stopbit: 1
- Overførelseshastighed: 1200bps
- Nødvendige ledere i RS-232: 3 stk. (GND, CTS og RXD)

C TS2- monitor

For at en mikrodatamat skal kunne fungere efter hensigten, skal der installeres et operativsystem. Operativsystemet gør det muligt for brugeren, at anvende systemet til at udføre de ønskede opgaver såsom uploade programmer ned i RAM og herefter udføre dem.
Et primitivt operativsystem kaldes en monitor, hvor monitoren der gøres brug af i projektet hedder TS2-monitor. TS2-monitoren stiller nogle krav til mikrodatamatsystemet:

Krav

- For seriel forbindelse bruges ACIA 6850 (UART)
- Adressedekoder til hukommelse, I/O og ACIA
- Reset-switch
- NMI-switch til interrupt 7

Krav til adresser	Adresseområder
EPROM eller FLASH-RAM	000000h - 001200h
TS2-monitor RAM placering	040000h - 040CFFh
Interrupt/exception vektorer flyttes til TS2-monitor RAM	040000h - 0407FFh
ACIA (UART)	800001h - 800003h

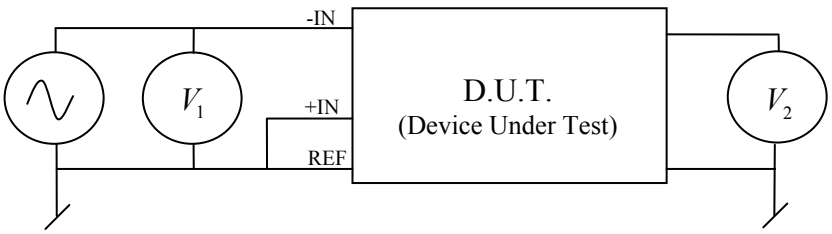
Tabel C.1: Krav til adresser for TS2-monitor

Monitoren indeholder en debugger, der kan bruges til fejlfinding af brugerens egne programmer. Dette gøres ved at monitoren udlæser processorens interne registre, og omsætter dem til ”beskeder” som brugeren kan forstå, og derved anvende dem til fejlfinding.

D Forstærkning/Galvanisk adskillelse

D.1 Test

D.1.1 Instrumenteringsforstærker

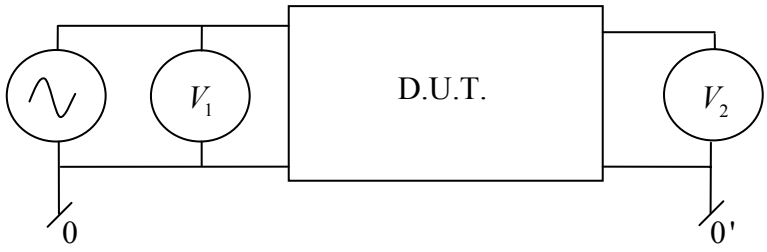
Formål	At undersøge instrumenteringsforstærkernes forstærkningsgrad.
Opstilling	
Instrumenter	• Voltmeter: Fluke 189 (l.b.nr. 52834) • Funktionsgenerator: B&O RC-Oscillator TG7 (l.b.nr. 07995) • Spændingsforsyning: B&O SN16 (l.b.nr. 33051)
Fremgangsmåde	På indgangen af instrumenteringsforstærkeren påtrykkes en sinusformet spænding på $1m\hat{V}$ (V_1) vha. en spændingsdeling og funktionsgenerator. Derpå måles udgangsspændingen og forstærkningen findes for alle kanaler.

Resultater

Kanal	R_G	V_1	V_2	V_2/V_1	Ønsket forstærkning	Afvigelse
1	100Ω	$1m\hat{V}$	$492m\hat{V}$	492	500	1,6%
2	100Ω	$1m\hat{V}$	$492m\hat{V}$	492	500	1,6%
3	100Ω	$1m\hat{V}$	$492m\hat{V}$	492	500	1,6%
4	100Ω	$1m\hat{V}$	$492m\hat{V}$	492	500	1,6%

Kommentar	Resultaterne må siges at være tilfredsstillende, idet afvigelsen fra det ønskede er minimal.

D.1.2 Galvanisk adskillelse og forstærkning

Formål	At undersøge hvorvidt forstærkningen og den galvaniske adskillelse lever op til kravet om linearitet. Desuden ønskes fastlagt maksimum og minimum grænser for forstærkning.
Opstilling	
Instrumenter	• Voltmeter: Fluke 189 (l.b.nr. 52834) • Funktionsgenerator: B&O RC-Oscillator TG7 (l.b.nr. 07995)

	• Spændingsforsyning: B&O SN16 (l.b.nr. 33051)
Fremgangsmåde	Indgangssiden forsynes med ±9V fra batteri, og udgangen forsynes med ±10V fra spændingsforsyningen. På indgangssiden påtrykkes et sinusformet signal på $348mV_{RMS}$ (målt med V_1) med frekvenser på henholdsvis 20, 100 og 150Hz. For hver frekvens stilles trimmemodstanden R-DIG (se figur 3.3) til 10, 60 og 110kΩ og udgangsspændingen måles. Dette foregår ved at måle trimmemodstanden med et ohmmeter inden hver måling. Der benyttes to forskellige nulpunkter, henholdsvis fra batteri og spændingsforsyning.
Resultater	Resultaterne for de fire kanaler ses i tabel D.2 til D.5.
Kommentar	I D.U.T. indgår kondensatoren C1, der tilhører instrumenteringsforstærkeren jf. figur 3.2. Der kan ud fra resultaterne ses , at kravet om en maksimal uliniearitet på ±3dB er opfyldt. Endvidere ses, at der kan opnås forstærkninger mellem ca. 0,8 og ca. 11,5, hvilket er tilfredsstillende.

Udledning af resultater

Resultaterne i tabel D.2 til D.5 er udregnet jf. eksemplet i tabel D.1 nedenfor. Beregningerne er foretaget på baggrund af en middel K3-værdi på 1,12 og et indgangssignal på $348mV_{RMS}$.

	Ønsket	**Målt**
Forstærkning	$G_{ønsket} = K3 \cdot \left(\frac{R_2}{R_1}\right) = 1{,}12 \cdot \left(\frac{10k\Omega}{10k\Omega}\right) = 1{,}12$	$G_{målt} = \frac{V_2}{V_1} = \frac{279mV_{RMS}}{348mV_{RMS}} = 0{,}80$
Spænding	$V_2 = V_1 \cdot G_{ønsket} = 348mV_{RMS} \cdot 1{,}12 = 390mV_{RMS}$	$V_2 = 279mV_{RMS}$
Afvigelse	$Afvigelse = 20 \cdot \log\left(\frac{G_{målt}}{G_{ønsket}}\right) = 20 \cdot \log\left(\frac{0{,}8}{1{,}12}\right) = -2{,}9dB$	

Tabel D.1: Eksempel på udregning af forstærkning, spænding og afvigelse for kanal 1 ved 20Hz og $R_2 = 10k\Omega$.

Kanal 1	**R2's modstandsværdi**		
Frekvens	**10kΩ**	**60kΩ**	**110kΩ**
20Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 0,80	6,70 / 5,16	12,28 / 9,26
- **Spænding**	390mV / 279mV	2338mV / 1796mV	4287mV / 3223mV
- **Afvigelse**	-2,9dB	-2,3dB	-2,5dB
100Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 1,05	6,70 / 6,10	12,28 / 11,48
- **Spænding**	390mV / 366mV	2338mV / 2128mV	4287mV / 4007mV
- **Afvigelse**	-0,6dB	-0,8dB	-0,6dB
150Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 1,06	6,70 / 6,22	12,28 / 11,82
- **Spænding**	390mV / 369mV	2338mV / 2170mV	4287mV / 4126mV
- **Afvigelse**	-0,5dB	-0,6dB	-0,3dB

Tabel D.2: Ønskede og målte resultater for kanal 1

Kanal 2	R2's modstandsværdi		
Frekvens	**10kΩ**	**60kΩ**	**110kΩ**
20Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 0,80	6,70 / 5,03	12,28 / 9,35
- **Spænding**	390mV / 279mV	2338mV / 1755mV	4287mV / 3264mV
- **Afvigelse**	-2,9dB	-2,5dB	-2,4dB
100Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 1,03	6,70 / 6,32	12,28 / 11,33
- **Spænding**	390mV / 359mV	2338mV / 2205mV	4287mV / 3955mV
- **Afvigelse**	-0,7dB	-0,5dB	-0,7dB
150Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 1,05	6,70 / 6,27	12,28 / 11,45
- **Spænding**	390mV / 365mV	2338mV / 2188mV	4287mV / 3997mV
- **Afvigelse**	-0,6dB	-0,6dB	-0,6dB

Tabel D.3: Ønskede og målte resultater for kanal 2

Kanal 3	R2's modstandsværdi		
Frekvens	**10kΩ**	**60kΩ**	**110kΩ**
20Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 0,84	6,70 / 5,03	12,28 / 9,28
- **Spænding**	390mV / 293mV	2338mV / 1755mV	4287mV / 3240mV
- **Afvigelse**	-2,5dB	-2,5dB	-2,4dB
100Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 1,04	6,70 / 6,37	12,28 / 11,33
- **Spænding**	390mV / 362mV	2338mV / 2223mV	4287mV / 3955mV
- **Afvigelse**	-0,6dB	-0,4dB	-0,7dB
150Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 1,05	6,70 / 6,44	12,28 / 11,34
- **Spænding**	390mV / 366mV	2338mV / 2247mV	4287mV / 3959mV
- **Afvigelse**	-0,6dB	-0,3dB	-0,7dB

Tabel D.4: Ønskede og målte resultater for kanal 3

Kanal 4	R2's modstandsværdi		
Frekvens	**10kΩ**	**60kΩ**	**110kΩ**
20Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 0,82	6,70 / 5,08	12,28 / 9,06
- **Spænding**	390mV / 286mV	2338mV / 1773mV	4287mV / 3163mV
- **Afvigelse**	-2,7dB	-2,4dB	-2,6dB
100Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 1,05	6,70 / 6,15	12,28 / 11,39
- **Spænding**	390mV / 366mV	2338mV / 2146mV	4287mV / 3976mV
- **Afvigelse**	-0,6dB	-0,7dB	-0,7dB
150Hz	(Ønsket/målt)	(Ønsket/målt)	(Ønsket/målt)
- **Forstærkning**	1,12 / 1,06	6,70 / 6,25	12,28 / 11,70
- **Spænding**	390mV / 369mV	2338mV / 2181mV	4287mV / 4085mV
- **Afvigelse**	-0,5dB	-0,6dB	-0,4dB

Tabel D.5: Ønskede og målte resultater for kanal 4

D.2 Worst-case beregninger

I det følgende ønskes forstærkningen ved 20Hz, med henholdsvis største og mindste K3-værdi (1,051/1,181) og kondensatorværdi $\left(1\mu F \pm 20\% = 0,8\mu F / 1,2\mu F\right)$, fastlagt. Resultaterne fremgår af tabel D.6.

Dæmpningen forårsaget af kondensatoren kan ses i ligning D.1.

$$G = \frac{R \cdot C \cdot \omega}{\sqrt{R^2 \cdot C^2 \cdot \omega^2 + 1}} \Rightarrow$$

$$G_{20Hz,\min} = \frac{10k\Omega \cdot 0,8\mu F \cdot 2\pi \cdot 20Hz}{\sqrt{\left(10k\Omega\right)^2 \cdot \left(0,8\mu F\right)^2 \cdot \left(2\pi \cdot 20Hz\right)^2 + 1}} = 0,709 = -2,99dB \qquad (D.1)$$

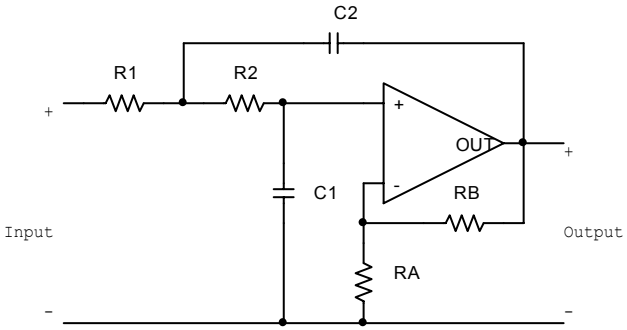
$$G_{20Hz,\max} = \frac{10k\Omega \cdot 1,2\mu F \cdot 2\pi \cdot 20Hz}{\sqrt{\left(10k\Omega\right)^2 \cdot \left(1,2\mu F\right)^2 \cdot \left(2\pi \cdot 20Hz\right)^2 + 1}} = 0,833 = -1,58dB$$

	Max	**Min**
Forstærkning	$G_{\max} = 0,833 \cdot 1,181 = 0,98$	$G_{\min} = 0,709 \cdot 1,051 = 0,75$
Spænding	$V_2 = 348mV_{RMS} \cdot 0,98 = 341mV_{RMS}$	$V_2 = 348mV_{RMS} \cdot 0,75 = 261mV_{RMS}$
Afvigelse	$Afvigelse_{\max} = 20 \cdot \log\left(\frac{0,98}{1,12}\right) = -1,16dB$	$Afvigelse_{\min} = 20 \cdot \log\left(\frac{0,75}{1,12}\right) = -3,48dB$

Tabel D.6: Worst-case beregninger for en kanal ved en frekvens på 20Hz og $R_2 = 10k\Omega$.

E Antialiasing filter

Det er valgt at benytte et Sallen & Key filter, se figur E.1, til realiseringen af såvel højpasfiltret som lavpasfiltret.



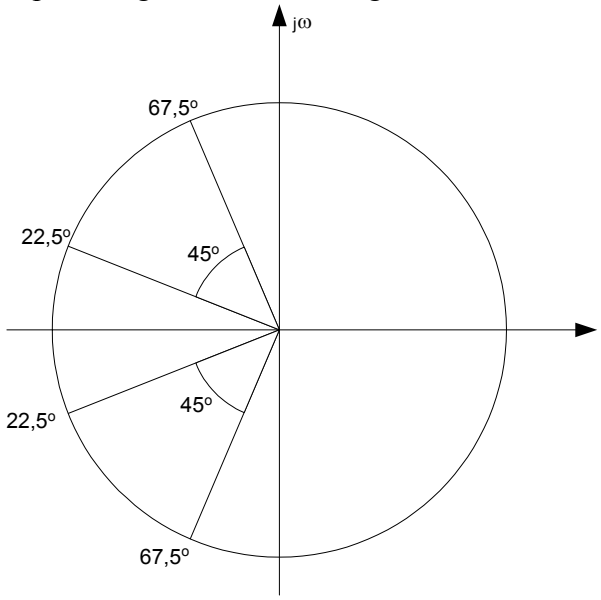
Figur E.1: Diagram over Sallen & Key 2.ordens filter[Filter s.172]

Denne type filter har mulighed for en noninverterende forstærkning, men da en noninverterende forstærkning ikke vil blive benyttet i det aktuelle tilfælde, forbindes minusbenet på filterets operationsforstærker direkte til udgangen.
Dette medfører at R_A vil have værdien ∞ og R_B er 0. Begge modstande kan derfor undlades.
Indsat i ligning E.1 ses, at forstærkningen i denne situation er 1.

$$A = 1 + \frac{R_B}{R_A} = 1 + \frac{0}{\infty} = 1 \qquad (E.1)$$

E.1 Lavpasfilter

Til beregning af komponentstørrelser benyttes teorien bag Butterworth lavpasfiltre [Filter s.157], da den matematiske model for denne type filtre kan implementeres i et Sallen & Key filter. Et sådant 4. ordens filter har 4 poler med placeringerne som vist i figur E.2.



Figur E.2: Polplacering for 4. ordens Butterworth lavpas filter[Filter s.164]

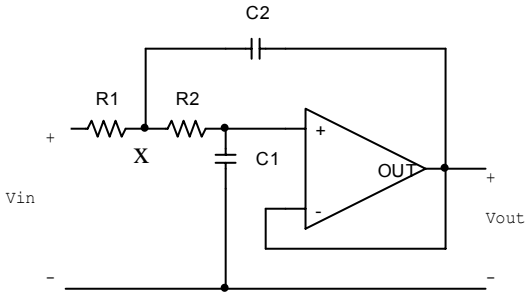
Kvalitetsfaktoren Q kan på baggrund af figur E.2 beregnes med ligning E.2:

$$Q = \frac{1}{2\cos(\psi)} \tag{E.2}$$

Dette giver Q værdierne 0,54 og 1,31, idet ψ har værdierne $\pm 22,5^{o}$ og $\pm 67,5^{o}$ jf. figur E.2.
For at benytte disse værdier skal overføringsfunktionerne for Butterworth modellen og Sallen & Key kredsløbet sammenlignes.
Overføringsfunktionen for Butterworth modellen ses i ligning E.3.

$$H(s) = \frac{K}{s^2 + s(\frac{\omega_0}{Q}) + {\omega_0}^2} \tag{E.3}$$

Overføringsfunktionen for Sallen & Key 2. ordens filter vil nu blive udledt.
Diagrammet for Sallen & Key lavpasfiltret er vist på figur E.3:



Figur E.3: Diagram for Sallen & Key lavpasfilter

Til at opskrive overføringsfunktionen benyttes KCL i knudepunktet (x) mellem R_1 og R_2 samt knudepunktet (out) mellem R_2 og C_1, derved fås:

$$KCL(x): \frac{V_{in} - V_x}{R_1} + (V_{out} - V_x) \cdot C_2 \cdot s + \frac{V_{out} - V_x}{R_2} = 0 \tag{E.4}$$

samt

$$KCL(out): \frac{V_x - V_{out}}{R_2} + (0 - V_{out}) \cdot C_1 \cdot s = 0 \tag{E.5}$$

Ved at isolere V_x i ligning E.5 og indsætte V_x i ligning E.4, fås følgende overføringsfunktion for lavpasfiltret:

$$H(s) = \frac{V_{out}}{V_{In}} = \frac{\frac{1}{R_1 R_2 C_1 C_2}}{s^2 + s\left(\frac{1}{R_1 C_2} + \frac{1}{R_2 C_2}\right) + \frac{1}{R_1 R_2 C_1 C_2}} \tag{E.6}$$

Sammenholdes denne overføringsfunktion med Butterworth modellen i ligning E.3 kan følgende ligning opskrives:

$$\frac{\omega_0}{Q} = \frac{1}{R_1 C_2} + \frac{1}{R_2 C_2} \qquad (E.7)$$

Derudover sættes $R_1 = R_2 = R$ [Filter s.181], hvilket giver denne reducerede ligning:

$$\frac{\omega_0}{Q} = \frac{2}{RC_2} \qquad (E.8)$$

Herefter kan komponentværdierne for en Q-værdi på Q = 0,54 beregnes.
For at kunne beregne C_2 sættes modstanden R til standardværdien 7,68kΩ og denne værdi indsættes i ligning E.8:

$$C_2 = \frac{2Q}{R\omega_0} = \frac{2 \cdot 0,54}{7680\Omega \cdot 2 \cdot \pi \cdot 150Hz} = 150nF \qquad (E.9)$$

Forholdet mellem C_1 og C_2 er beskrevet som følgende: [Filter s. 174]

$$C_2 = 2Q \; og \; C_1 = \frac{1}{2Q} \Rightarrow \frac{C_1}{C_2} = \frac{1}{4Q^2} \qquad (E.10)$$

Af dette kan kondensatoren C_1 beregnes:

$$C_1 = C_2 \frac{1}{4 \cdot Q^2} = 150nF \frac{1}{4 \cdot 0,54^2} = 128nF \qquad (E.11)$$

Denne værdi sættes til standardværdien 122nF.
Ligeledes beregnes komponentværdierne for Q-værdien Q = 1,31.
Modstanden R sættes til 41,2kΩ, og denne værdi indsættes i ligning E.8, hvor C_2 udbyttes med C_4.

$$C_4 = \frac{2Q}{R\omega_0} = \frac{2 \cdot 1,31}{41200\Omega \cdot 2 \cdot \pi \cdot 150Hz} = 68nF \qquad (E.12)$$

Forholdet imellem kondensatorerne C_3 og C_4 er identisk med forholdet imellem kondensatorerne C_1 og C_2, udregnet i ligning E.11, hvorfor C_3 har følgende værdi:

$$C_3 = C_4 \frac{1}{4 \cdot Q^2} = 68nF \frac{1}{4 \cdot 1,31^2} = 10nF \qquad (E.13)$$

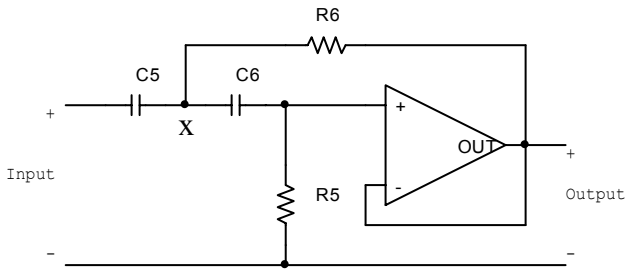
Dermed er komponentværdierne for lavpasfiltret bestemt.

E.2 Højpasfilter

Til realisering af højpasfiltret benyttes den generelle overføringsfunktion for et 2. ordens højpasfilter [Filter s.172]:

$$H(s) = \frac{Ks^2}{s^2 + s(\frac{\omega_0}{Q}) + {\omega_0}^2} \qquad (E.14)$$

Til realisering af denne overføringsfunktion vil filteret i figur E.4 blive benyttet.



Figur E.4: Diagram for Butterworth højpasfilter[Filter s. 181]

Til opstilling af overføringsfunktionen benyttes KCL i knudepunktet (x) imellem C_5 og C_6 og knudepunktet (out) imellem C_6 og R_6, derved fås:

$$KCL(x) : (V_{in} - V_x) \cdot C_5 \cdot s + \frac{V_{out} - V_x}{R_6} + (V_{out} - V_x) \cdot C_6 \cdot s = 0 \qquad (E.15)$$

Samt

$$KCL(out) : (V_x - V_{out}) \cdot C_6 \cdot s + \frac{0 - V_{out}}{R_5} = 0 \qquad (E.16)$$

Ved at isolere V_x i ligning E.16 og indsætte denne i ligning E.15 kan overføringsfunktionen for højpasfiltret opstilles:

$$H(s) = \frac{V_{out}}{V_{in}} = \frac{s^2}{s^2 + s(\frac{1}{C_5 R_5} + \frac{1}{C_6 R_5}) + \frac{1}{R_5 R_6 C_5 C_6}} \qquad (E.17)$$

Sammenholdes denne overføringsfunktion med den generelle model for et 2. ordens højpasfilter jf. ligning E.14, kan følgende ligning opskrives:

$$\frac{\omega_0}{Q} = \frac{1}{C_5 R_5} + \frac{1}{C_6 R_5} \qquad (E.18)$$

Derudover sættes $C_5 = C_6 = C$ [Filter s.181]. Dette giver denne reducerede ligning:

$$\frac{\omega_0}{Q} = \frac{2}{CR_5} \qquad (E.19)$$

Knækfrekvensen ω_0 er $20Hz \cdot 2 \cdot \pi \approx 125,7\, rad/sek$, og Q-værdien for et 2.ordensfilter er $\sqrt{2}/2$ [Filter s.163]. Kondensatorværdien sættes til 47nF, hvorefter R_5 beregnes:

$$R_5 = \frac{2Q}{\omega_0 C} = \frac{2 \cdot \sqrt{2}/2}{20Hz \cdot 2 \cdot \pi \cdot 47nF} = 239k\Omega \qquad (E.20)$$

Forholdet mellem modstandene R_5 og R_6 er beskrevet som følgende [Filter s.181]:

$$R_5 = \frac{1}{2Q}\ og\ R_6 = 2Q \Rightarrow \frac{R_6}{R_5} = 4Q^2 \qquad (E.21)$$

Af dette kan modstanden R_6 beregnes:

$$R_6 = R_5 \cdot 4 \cdot Q^2 = 239k\Omega \cdot 4 \cdot 0,707^2 = 478k\Omega$$

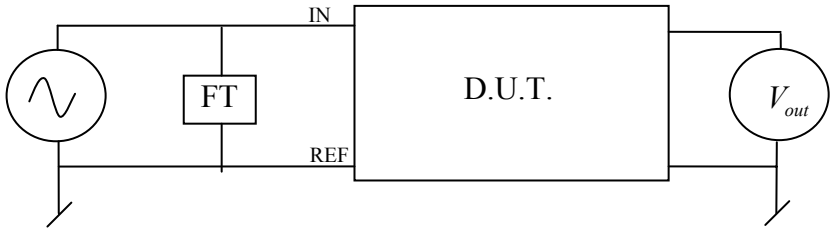
Højpasfiltrets komponentværdier er således bestemt.

E.3 Test af filter

Formål

At undersøge knækfrekvenser og dæmpning af signal ved 500 Hz.

Opstilling



Instrumenter

- Voltmeter: Fluke 37< (l.b.nr. 08514)
- Funktionsgenerator: B&O RC-Oscillator TG7 (l.b.nr. 07995)
- Spændingsforsyning: B&O SN16 (l.b.nr. 33051)
- Frekvenstæller: PM 6667 (FT) (l.b.nr. 07900)

Fremgangsmåde

Test 1: Ved undersøgelse af knækfrekvenser blev der brugt en indgangsspænding på 1V. Derefter blev frekvensen indstillet således V_{out} blev 3 dB mindre.

Test 2: Ved undersøgelse af signalet ved 500 Hz blev der ligeledes brugt en indgangsspænding på 1V. Frekvensen blev indstillet til 500 Hz hvorefter spændingen blev aflæst.

Resultater

Test 1	Nedre 3 dB	Øvre 3 dB	Forventet nedre frekvens	Forventet øvre frekvens	Nedre 3 dB afvigelse	Øvre 3 dB afvigelse
1	23,5	155	20	150	13%	3%
2	22,8	149	20	150	12%	-1%
3	23,9	154	20	150	16%	3%
4	22,7	152	20	150	12%	1%

Test 2	Signal ved 500 Hz	Dæmpning i dB	Forventet dæmpning i dB	Afvigelse
1	0,008	-41,9	-41,8	0,2%
2	0,007	-43,1	-41,8	3%
3	0,008	-41,9	-41,8	0,2%
4	0,008	-41,9	-41,8	0,2%

Kommentar

Resultaterne må siges at være tilfredsstillende, da frekvenserne er acceptable for båndpasset. Ved test 2 kan målingerne være lidt upræcise på grund af de lave spændinger.

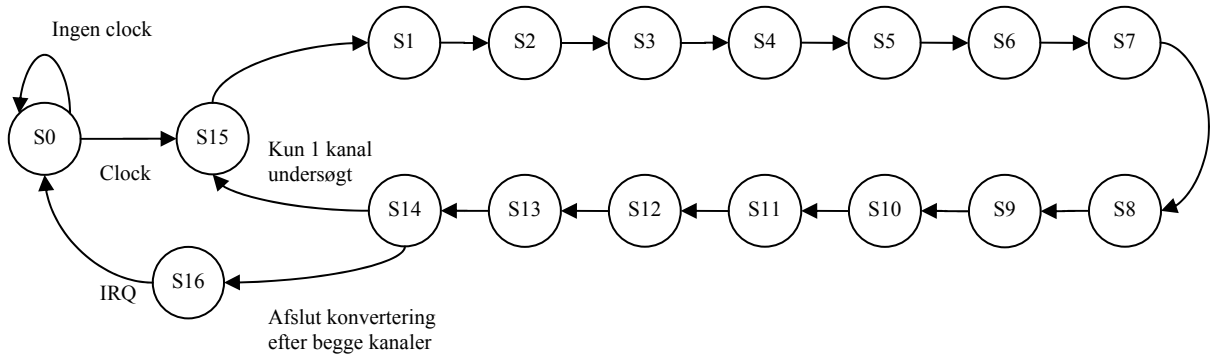
F Multiplexer og A/D-konverter

F.1 State tabel

State	Næste state	Beskrivelse
0	Hvis CS lav, så State 15. Ellers State 0	Neutral
1	State 2	Startbit til A/D-konverter set-up data
2	State 3	Valg af enkelt signaler på A/D-konverter
3	State 4	Valg af lige multiplexer kanaler
4	State 5	Høj bit til valg af multiplexer kanal
5	State 6	Lav bit til valg af multiplexer kanal
6	State 7	Startbit kommer fra A/D-konverter
7	State 8	Udlæsning af MSB fra A/D-konvertering
8	State 9	Udlæsning af bit 6 fra A/D-konvertering
9	State 10	Udlæsning af bit 5 fra A/D-konvertering
10	State 11	Udlæsning af bit 4 fra A/D-konvertering
11	State 12	Udlæsning af bit 3 fra A/D-konvertering
12	State 13	Udlæsning af bit 2 fra A/D-konvertering
13	State 14	Udlæsning af bit 1 fra A/D-konvertering
14	Hvis 1. konvertering så State 15, ellers State 16	Udlæsning af bit 0 fra A/D-konvertering
15	State 1	Pause ved konvertering
16	State 0	Afsendelse af interrupt
17-31	State 0	Ikke aktive interrupts

Tabel F.1: State tabel over konvertering af to kanaler

F.2 State diagram



Figur F.1: State diagram over konvertering af to kanaler

F.3 Kode til PA7540

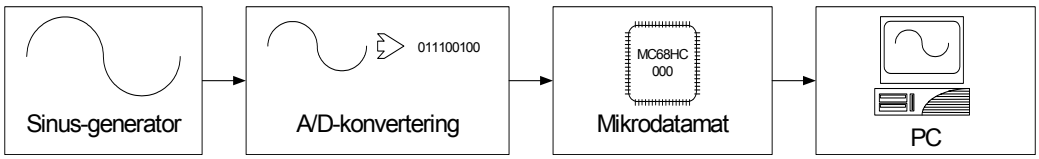
```
Title 'A/D konverter interface og adressedekoder'
Designer 'Gruppe E413'
Date '26-04-2002'

Description
Giver nødvendigt signal til A/D-konverter, så den kan svare på forespørgsel fra CPU.

Input:
CLK1          PIN 2:    400kHz clock. Tages fra CLK400K udgangen.
CLK2          PIN 16:   800kHz clock. Kommer fra E-benet på M68k.
CLK2K         PIN 5:    2kHz samplings clock.


AS            PIN 6:    Adresse strobe fra M68k. Angiver hvornår der er et gyldigt signal
                        på adressebussen.
A18           PIN 9:    A18 fra M68k.
A19           PIN 10:   A19 fra M68k.
A23           PIN 11:   A23 fra M68k.
VMA           PIN 19:   VMA fra M68k.

Output:
CLK400K       PIN  3:   400kHz clock signal til A/D-konverteren og interne registre.
                        Forbindes eksternt til CLK1 og ADC838.
CS_ADC        PIN 26:   Chipselect til A/D-konverteren.
ADC_DATA      PIN 27:   Set-up data til ADC.
SR_CLK        PIN 17:   Clocksignal til skifteregistre.
IRQ1          PIN 12:   Interruptsignal for færdig konvertering på kanal 1 og 2.
IRQ2          PIN 13:   Interruptsignal for færdig konvertering på kanal 3 og 4.


CS_BUF        PIN 4:    Chipselect signal til bufferkredse for A/D-konverter.
CS_ROM        PIN 21:   CS til ROM.
CS_RAM        PIN 20:   CS til RAM.
CS_ACIA       PIN 17:   CS til ACIA.
DTACK         PIN 7:    Data Acknowledge til M68k.
VPA           PIN 18:   VPA til M68k.

End_Desc;

PA7540

CLK1 PIN 1 "400kHz clock. Tages fra CLK400K udgangen.
CLK2 PIN 13 "800kHz clock. Kommer fra E-benet på M68k.

"IOC CONFIGURATION DECLARATION
"IOC (PIN_NO 'PIN_NAME'          POLARITY     IO_TYPE      ) <- LCC_NO

IOC ( 2      'CLK400K'           Pos          IO           ) <- 3D
IOC ( 3      'CS_BUF'            Neg          Out          ) <- 1B
IOC ( 4      'CLK2K'             Pos          IO           ) <- 3A
IOC ( 5      'AS'                Pos          IO           ) <- 1D
IOC ( 6      'DTACK'             Neg          Out          ) <- 5A
IOC ( 7      'A18'               Pos          InCom        ) <- 2A
IOC ( 8      'A19'               Pos          InCom        ) <- 2B
IOC ( 9      'A23'               Pos          InCom        ) <- 3B
IOC (10      'IRQ1'              Neg          Out          ) <- 4B
IOC (11      'IRQ2'              Neg          Out          ) <- 5B
IOC (14      'SR_CLK'            Pos          Out          ) <- 1C
IOC (15      'VPA'               Neg          Out          ) <- 2C
IOC (16      'VMA'               Pos          InCom        ) <- 3C
IOC (17      'CS_ACIA'           Neg          Out          ) <- 4C
IOC (18      'CS_KAL'            Neg          Out          ) <- 5C
IOC (19      'A20'               Neg          InCom        ) <- 4A
```

```
IOC (20       'CS_RAM'            Neg          Out          ) <- 2D
IOC (21       'CS_ROM'            Neg          Out          ) <- 1A
IOC (22       'CS_ADC'            Neg          Out          ) <- 4D
IOC (23       'ADC_DATA'          Pos          IO           ) <- 5D

"LCC CONFIGURATION DECLARATION
"LCC (LCC_NO  'LCC_NAME'          REG_TYPE     CLK_TYPE     BUR_OUT      EXT_OUT     )

LCC (1A       ''                  D            SumD         Reg          SumB        )
LCC (2A       'CH0'               D            SumD         Reg          SumB        )
LCC (3A       'CH1'               D            SumD         Reg          Reg         )
LCC (4A       ''                  D            Neg          Reg          SumB        )
LCC (5A       'CLKDIV'            D            SumC         Reg          SumB        )
LCC (1B       'T0'                D            Neg          Reg          SumB        )
LCC (2B       'T1'                D            Neg          Reg          Reg         )
LCC (3B       'T2'                D            Neg          Reg          Reg         )
LCC (4B       'T3'                D            Neg          Reg          SumB        )
LCC (5B       'T4'                D            Neg          Reg          SumB        )
LCC (1C       'TRIG'              JK           Pos          Reg          SumC        )
LCC (2C       ''                  D            SumC         Reg          SumB        )
LCC (3C       ''                  D            SumC         Reg          SumB        )
LCC (4C       ''                  D            SumC         Reg          SumB        )
LCC (5C       ''                  D            SumC         Reg          SumB        )
LCC (1D       ''                  D            SumD         Reg          SumB        )
LCC (2D       ''                  D            SumC         Reg          SumB        )
LCC (3D       ''                  D            SumC         Reg          SumB        )
LCC (4D       ''                  D            SumC         Reg          SumB        )
LCC (5D       ''                  D            SumC         Reg          SumB        )

"GBC CONFIGURATION DECLARATION
"GBC (GROUP   LCC_CLK   IOC_CLK   )

Global = 1
GBC (A        CLK1      CLK1      )
GBC (B        CLK1      CLK1      )


DEFINE
TIMER = [T4 T3 T2 T1 T0]


STATE_DIAGRAM TIMER
"TIMER MASKINE. DEFINERER OVERORDNET TIMING.
STATE 0:      IF CLK2K & TRIG     THEN 15      "Timeren er lavet med 32 states. State 0
                                  ELSE 0       "er neutralt state. Proceduren starter når
                                               "både CLK2K og TRIG er
STATE 1:      GOTO 2                           "høje. TRIG går høj når CLK2K har været
STATE 2:      GOTO 3                           "lav. TRIG resettes ved afslutningen af
STATE 3:      GOTO 4                           "hele konverteringscyklusen.
STATE 4:      GOTO 5
STATE 5:      GOTO 6
STATE 6:      GOTO 7
STATE 7:      GOTO 8
STATE 8:      GOTO 9
STATE 9:      GOTO 10
STATE 10:     GOTO 11
STATE 11:     GOTO 12
STATE 12:     GOTO 13
STATE 13:     GOTO 14
STATE 14:     IF CH0              "Test for om begge kanaler har været læst. Hvis ja,
              THEN 15             "så gåes i neutralt state.
              ELSE 16
STATE 15:     GOTO 1
STATE 16:     GOTO 0              "Interrupt state. Send interrupt request.
STATE 17:     GOTO 0
```

```
STATE 18:     GOTO 0
STATE 19:     GOTO 0
STATE 20:     GOTO 0
STATE 21:     GOTO 0
STATE 22:     GOTO 0
STATE 23:     GOTO 0
STATE 24:     GOTO 0
STATE 25:     GOTO 0
STATE 26:     GOTO 0
STATE 27:     GOTO 0
STATE 28:     GOTO 0
STATE 29:     GOTO 0
STATE 30:     GOTO 0
STATE 31:     GOTO 0
END;

EQUATIONS

"Internal or External output names appended with extensions:
"  1) .COM for Combinatorial Internal or External Output (Sum A, B, or C)
"  2) .OE for External Output Enable (Sum D only)
"  3) .CLK for Asynchronous Logic Control Clock (Sum C or D)
"  4) .AP for Asynchronous Preset for LCC Register (Sum B only)
"  5) .AR for Asynchronous Reset for LCC Register (Sum C only)


"Kontrol af A/D-konvertering:

TRIG.J = !CLK2K;                                "Skifteregister TRIG aktiveres når
                                                "samplingsclocken går lav. Bruges til at
                                                "trigge på samplingsclockens forkant.
TRIG.K = !T4 & T3 & T2 & T1 & !T0 & !CH0;       "TRIG resettes når konverterings-
                                                "proceduren er færdig.

"ADC_DATA bruges til at læse set-up data ind i A/D-konverteren
ADC_DATA.COM=!T4 & !T3 & !T2 & !T1 & T0 + !T4 & !T3 & !T2 & T1 & !T0 +
             !T4 & !T3 & T2 & !T1 & !T0 & CH1 + !T4 & !T3 & T2 & !T1 & T0 & CH0

"Definering af ADC_DATA som udgang.
ADC_DATA.OE =!T4 & !T3 & !T2 & T1 + !T4 & !T3 & !T2 & T0 + !T4 & !T3 & T2 & !T1;

"Giver chipselect til A/D-konverteren når TIMER giver signal til det.
CS_ADC.COM =  !T4 & !T3 & !T2 & !T1 & T0 + !T4 & !T3 & !T2 & T1 & !T0 +
              !T4 & !T3 & !T2 & T1 & T0 + !T4 & !T3 & T2 & !T1 & !T0 +
              !T4 & !T3 & T2 & !T1 & T0 + !T4 & !T3 & T2 & T1 & !T0 +
              !T4 & !T3 & T2 & T1 & T0 + !T4 & T3 & !T2 & !T1 & !T0 +
              !T4 & T3 & !T2 & !T1 & T0 + !T4 & T3 & !T2 & T1 & !T0 +
              !T4 & T3 & !T2 & T1 & T0 + !T4 & T3 & T2 & !T1 & !T0 +
              !T4 & T3 & T2 & !T1 & T0 + !T4 & T3 & T2 & T1 & !T0;
CS_ADC.OE =   1;

"Sender clocksignal til skifteregistre når der er data.
SR_CLK.COM =  (!T4 & !T3 & T2 & T1 & T0 + !T4 & T3 & !T2 & !T1 & !T0 +
              !T4 & T3 & !T2 & !T1 & T0 + !T4 & T3 & !T2 & T1 & !T0 +
              !T4 & T3 & !T2 & T1 & T0 + !T4 & T3 & T2 & !T1 & !T0 +
              !T4 & T3 & T2 & !T1 & T0 + !T4 & T3 & T2 & T1 & !T0) & CLK1;
SR_CLK.OE = 1;          "Aktiverer eksternt output fra ovenstående udtryk.

CLKDIV.D = !CLKDIV;     "Clockdeler. Deler clocken på 800kHz, der kommer fra
                        "enable benet på CPU'en ned til 400kHz, som er den største
CLKDIV.CLK = CLK2;      "frekvens, A/D-konverteren kan klare.

CLK400K.COM = !CLKDIV;            "Udgang fra clockdeleren.
CLK400K.OE = 1;
```

```
"CH0 er et internt register, der definerer hvilken kanal, der læses på.
CH0.D =       !T4 & T3 & T2 & !T1 & T0 & !CH0;
CH0.AR =      !T4 & !T3 & !T2 & !T1 & !T0;
CH0.CLK =     !T4 & T3 & T2 & !T1 & T0 & CLK1;

CH1.D =       !CH1;
CH1.CLK =     !CLK2K;

IRQ1.COM =    T4 & !T3 & !T2 & !T1 & !T0 & !CH1;            "Aktiver IRQ1 ved state 25,
IRQ1.OE = 1;                                                "når CH1 er lav.

IRQ2.COM = T4 & !T3 & !T2 & !T1 & !T0 & CH1;   "Aktiver IRQ2 ved state 25, når
IRQ2.OE = 1;                                   "CH1 er høj.

T0.D = 0;               "TMR0-4 er defineret i statediagrammet TIMER, og bruges ikke her.
T0.AR = 0;

T1.D = 0;
T1.AR = 0;

T2.D = 0;
T2.AR = 0;

T3.D = 0;
T3.AR = 0;

T4.D = 0;
T4.AR = 0;
```

F.4 Test af A/D-konvertering

Formål	At kontrollere om et analog sinussignal gennem A/D-konvertering, kan aflæses af mikrodatamaten.
Opstilling	
Instrumenter	• Spændingsforsyning: Delta Elektronika E030–3 LBNR: 08500 Tage Juul Elektronic LBNR: 07379 • Funktionsgenerator: Wavetek Model 116 LBNR: 06924 • Oscilloskop: Agilent 54621A LBNR: 33865 • PC
Fremgangsmåde	• Der indlæses programkode i mikrodatamaten (appendiks I.4), som læser én kanals 6 sekunders sampling ud til PC'en, ved en samplingsfrekvens f_s på 1000 Hz. Koden er skrevet til at sample én kanal ad gangen. • Sinus signal sættes ind på den valgte kanal på A/D-konverteren. • Programmet initialiseres og overfører data for kanalen ud til PC'en. • Data behandles, og sammenlignes med det oprindelige sinus-signal, med en maksimal amplitude på 5 Vpp, som ses på figur F.2. • Punkt 1-4 gennemføres med flg. konfigurationer; 20 Hz og 100 Hz for kanal 1-4 én ad gangen.
Resultater	Resultaterne for de fire kanaler ses i figurerne F.3 til F.6. Som det ses i figurerne bliver inputtet gengivet ved udlæsning af data. Mere specifikke data kan ses på bilag [CD:\Test\testdata]

Følgende skema indeholder afmålte gennemsnitsværdier for sinussignalernes maksimum- og minimumspunkter. For den analoge del er disse maks-min-værdier udregnet til hvor stor en procentdel de udgør af de 5 volt, som er den referenceværdi, A/D-konverteren sampler i forhold til. Det samme gælder for de samplede værdier i forhold til 255, som er max for disse. Disse værdier er subtraheret fra hinanden for at udregne den procentvise afvigelse for hver kanal i forhold til det oprindelige analoge signal. De skønsmæssig minimum- og maksimumværdier er approksimeret udfra de målte signaler, hvor der er valgt at tage værdierne udfra 10 perioder af signalet, midt i målingen.

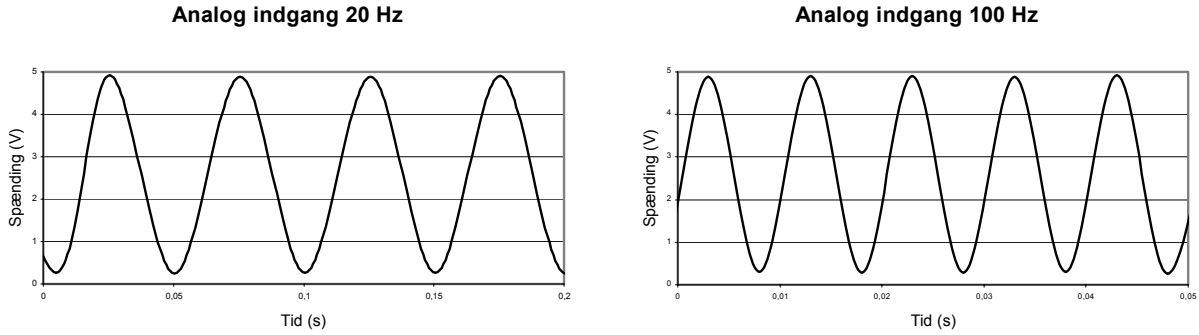
	Skønsmæssig minimum	Afvigelse	Skønsmæssig maksimum	Afvigelse
Analog 20 Hz	0,27 V		4,93 V	
Kanal 1 20 Hz	14	0,09 %	252	0,22 %
Kanal 2 20 Hz	14	0,09 %	251	-0,17 %
Kanal 3 20 Hz	13	-0,31 %	251	-0,17 %
Kanal 4 20 Hz	14	0,09 %	252	0,02 %
Analog 100 Hz	0,28 V		4,94 V	
Kanal 1 100 Hz	14	-0,11 %	253	0,41 %
Kanal 2 100 Hz	15	0,28 %	251	-0,37 %
Kanal 3 100 Hz	14	-0,11 %	252	0,02 %
Kanal 4 100 Hz	14	-0,11 %	252	0,02 %

Det kan aflæses at afvigelsen ved 20 Hz i minimumpunktet ligger mellem -0,31 og +0,09 procent. Og maksimumpunktet afviger med -0,17 og +0,22 procent. For 100 Hz målingerne er afvigelserne i minimumspunktet -0,11 og +0,28 procent. Og i maksimumpunktet -0,37 og + 0,41 procent.

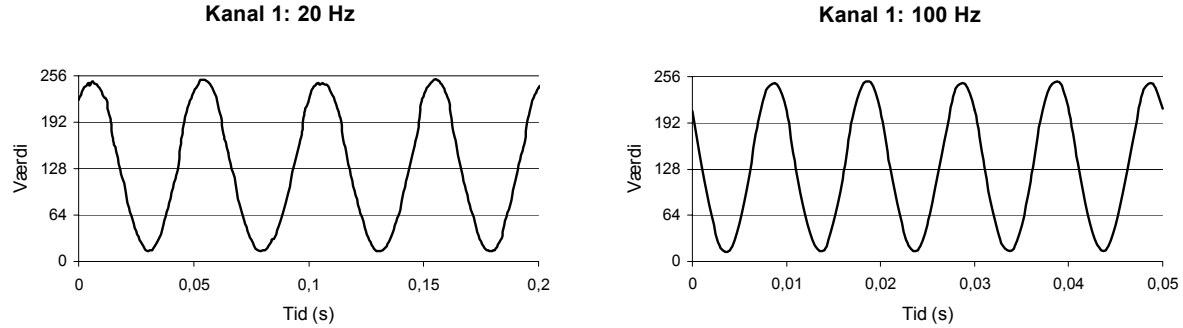
Kommentar

De procentvise afvigelser for hver digital kanal i forhold til den målte analoge kommer maximalt op på en afvigelse på 0,41%. Disse afvigelser er acceptable, da det ikke var muligt at tage målingerne på nøjagtig samme tid, og der vil derfor være afvigelser mellem hver måling.
Der kan ud fra disse resultater fastslås at det i A/D-konverteren indsendte signal, svarer til det fra mikrodatamaten udlæste data.
Det kan ses på figurerne F.3 til F.6 at der er afvigelser på 20 Hz målingerne, de er ikke perfekte sinus-kurver. Denne afvigelse kan forklares ved at A/D-konverteren støjer på den analoge forsyningsspænding. Dette er blevet målt før denne test og efter indgående fejlfinding blev det afhjulpet med en kondensator, men det er desværre ikke nok til at fjerne støjen helt, da denne var meget kraftig. Det er denne støj der har indvirkning på målingerne.
Grunden til at støjen har indvirkning, er at denne også har indflydelse på referencespændingen, som A/D-konverteren måler i forhold til. Derfor kan en spænding på A/D-konverterens indgang, måles til forskellige værdier på forskellige tidspunkter. Det er sandsynligt, at det er grunden til støjen kan ses på de målte kurver. Støjen er ved måling på referencespændingen bestemt til 100 mVpp, det svarer til at samplingværdierne max vil svinge med ±3 trin ud af de 255 mulige, i værste tilfælde.
Hvis der ses på målingerne i figurerne F.3 til F.6, kan det ses at støjen er tydeligere på 20 Hz målingerne end ved 100 Hz. Grunden til dette, er at der er flere målinger pr. periode ved 20 Hz, og at amplitude-ændringen pr. sample er

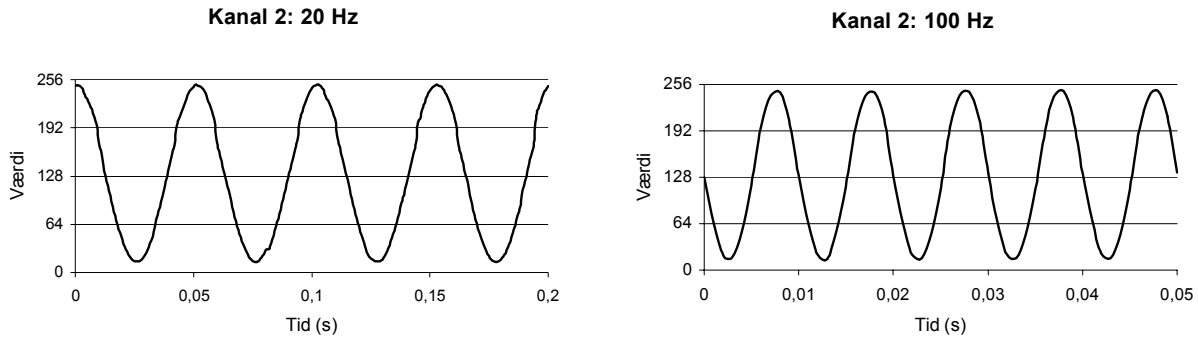
	mindre end ved 100 Hz, og støj dermed vil kunne få større synlig indvirkning på måleresultatet. Ved 100 Hz har denne støj også indvirkning, men er ikke ligeså tydelig på kurverne, da der for det første er længere mellem målepunkter pr. periode og de amplitudemæssige afvigelserne derfor ikke vil være lige så tydelige. På trods af at måleresultaterne ikke er perfekte, i forhold til signalet der blev samplet, er de dog for projektet tilfredsstillende. Det har i det aktuelle system ikke afgørende betydning. Signalets karakteristik ændres på grund af støjen, det er dog ikke selve signalet, der er afgørende i dette projekt, men signalets amplitude. Denne vil dog også være påvirket af støjen, men i forhold til et EMG-signal med fuld amplitude, vil denne støj have en minimal betydning. Det er derfor valgt, at arbejde videre med systemet, på trods af støjen.



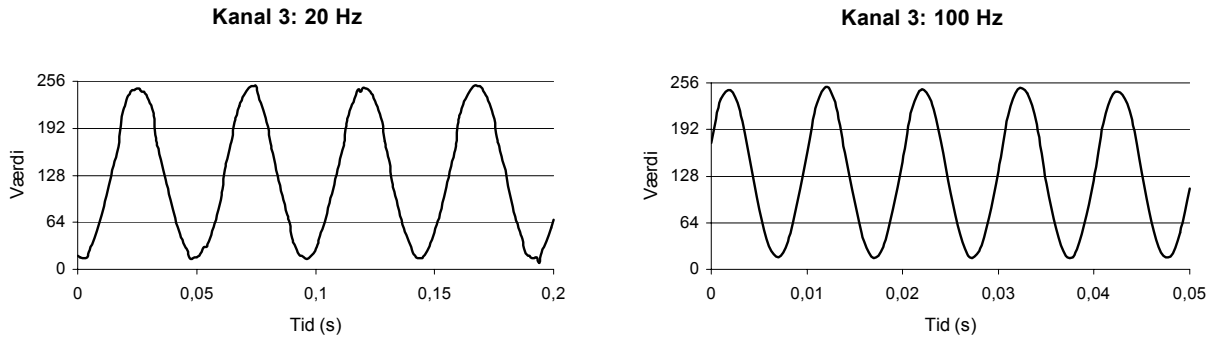
Figur F.2: Indgangssignal ved test af A/D-konverteringen



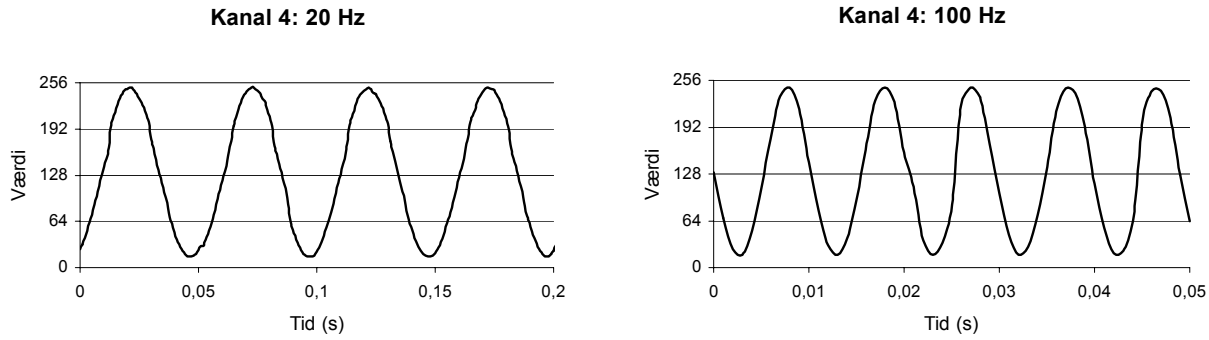
Figur F.3: Resultat på kanal 1



Figur F.4: Resultat på kanal 2



Figur F.5: Resultat på kanal 3

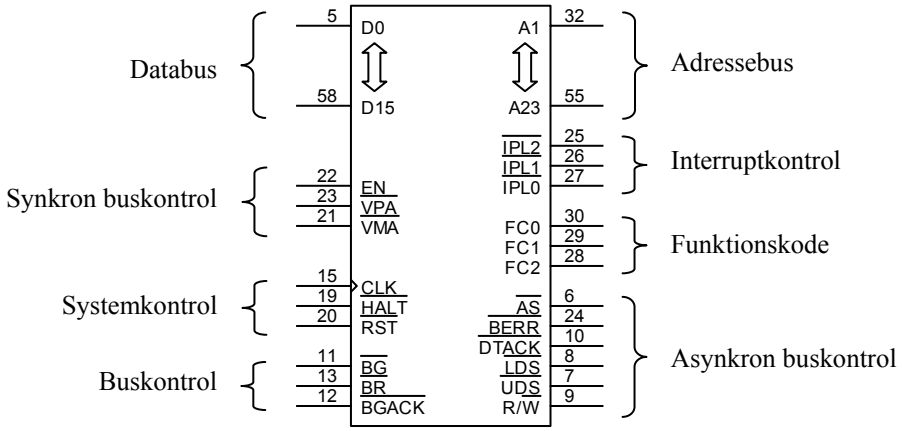


Figur F.6: Resultat på kanal 4

G Mikrodatamaten

G.1 CPU

G.1.1 Benbeskrivelser



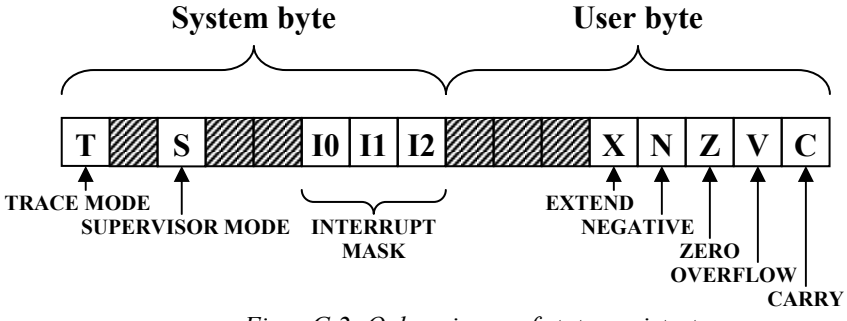
Figur G.1: Betegnelser og overordnet funktion for benene på M68000 processoren

Ben (antal)	Beskrivelse
Adressebus (23)	Tristate output der angiver, hvilken adresse CPU'en læser fra eller skriver til.
Databus (16)	Input og tristate output der af CPU'en bruges til at læse og skrive data fra den adresse, der er angivet af adressebussen.
Synkron buskontrol	*EN (Enable):* Clock på 1/10 af CPU'ens clockfrekvens. *VPA* (Valid Peripheral Address):* Input der sættes logisk høj, når en ekstern enhed ønsker synkron busadgang, hvilket anvendes af ACIA'en. *VMA* (Valid Memory Address):* Tristate output fra CPU'en der angiver om indholdet af adressebussen er korrekt.
Asynkron buskontrol	*AS* (Address Strobe):* Tristate output der angiver om indholdet af adressebussen er gyldig. *BERR* (Bus ERRor):* Input som indikerer fejl på bussen. *DTACK* (Data Transfer Acknowledge):* Input til CPU'en der sættes logisk høj af den enhed, som CPU'en forsøger at få adgang til, i det tilfælde at adgangen accepteres. *LDS* (Lower Data Strobe):* Tristate output der angiver om der skrives til eller læses fra den nedre del af et *Word*. *UDS* (Upper Data Strobe):* Tristate output der angiver om der skrives til eller læses fra den øvre del af et *Word*. *R/W* (Read/Write):* Angiver om CPU'en skal skrive til eller læse fra den pågældende adresse.

<table>
<tr><td>Funktionskode (3)</td><td>Tristate output der angiver typen af cyklus der udføres i CPU’en jf. nedenstående tabel:
<table>
<tr><th>FC2</th><th>FC1</th><th>FC0</th><th>Cyklus type</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Udefineret</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Bruger data</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Bruger program</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Udefineret</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Udefineret</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Supervisor data</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Supervisor program</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>CPU (interrupt)</td></tr>
</table></td></tr>
<tr><td>Interruptkontrol (3)</td><td>Interrupt indgange.</td></tr>
<tr><td>Systemkontrol</td><td>CLK (CLocK): TTL-input der bestemmer CPU’ens clockfrekvens.
HALT*: Input og open drain output. Sættes denne høj, vil CPU’en standse og alle tristate output vil gå i højimpedant tilstand. Denne egenskab kan bruges til at køre en enkelt buscyklus ad gangen, hvilket kan være hensigtsmæssigt ifbm. fejlfinding. Ved en fejl i CPU’en, som den ikke selv kan håndtere, vil HALT* gå logisk høj.
RST* (ReSeT): Input og open drain output, der som input anvendes til at resette CPU’en. Ved et reset skal RST* og HALT* holdes logisk høj i minimum 100ms. Eksekveres instruktionen <code>RESET</code> i CPU’en vil RST* gå høj i en periode på 124 clock cyklusser.</td></tr>
<tr><td>Buskontrol</td><td>BG* (Bus Grant): Output der giver eksterne enheder kontrol over bussen.
BR* (Bus Request): Input der angiver at en ekstern enhed ønsker kontrol over bussen.
BGACK* (Bus Grant ACKnowledge): Input der anvendes til overføring af kontrol med bussen.</td></tr>
</table>

Tabel G.1: Beskrivelse af mikroprocessorens ben

G.1.2 Statusregister i M68HC000



Figur G.2: Opbygningen af statusregistret

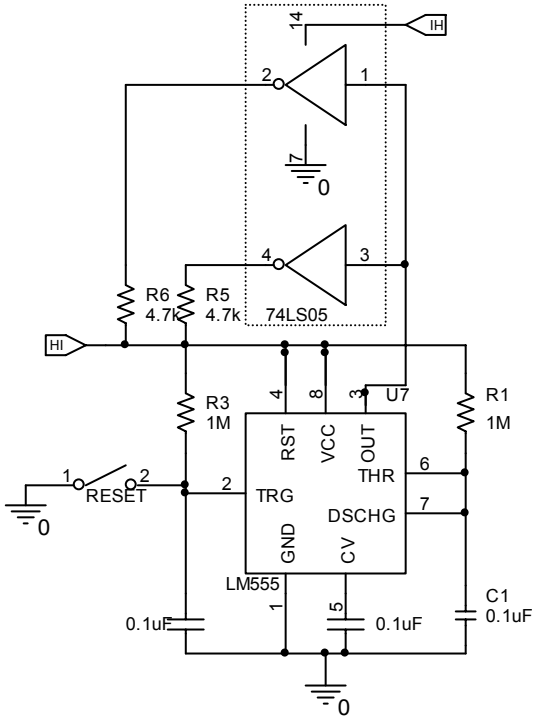
- **Trace Mode:** Indikerer at CPU’en standser efter udførelse af hver instruktion. Anvendes i forbindelse med fejlfinding.
- **Supervisor Mode:** Angiver om CPU’en befinder sig i ”Supervisor Mode”, hvilken har højere prioritet end ”User Mode”. Ved en exeption/et interrupt vil CPU’en automatisk gå i

denne tilstand, hvilket er den eneste måde, at gå fra User til Supervisor Mode. Visse instruktioner kan kun udføres i denne tilstand.

- **Interrupt Mask:** Angiver hvilke interrupts der er aktive jf. tabel 6.3.
- **Extend:** Et sandt 'mente' bit, der er 1 i de tilfælde, hvor en byte/et word/et longword bliver større end 8/16/32 bit.
- **Negative:** Angiver MSB bit og dermed fortegn for et register.
- **Zero:** Angiver at samtlige bit i registret er nul.
- **Overflow:** Er sat til 1 hvis resultatet af en aritmetisk operation fylder mere end der er plads til i registret.
- **Carry:** Angiver det sidste bit i forbindelse med aritmetiske operationer.

G.1.3 Power On Reset

Efter CPU'en forsynes med strøm, vil der gå op til 100ms inden den er i stand til at køre optimalt. Derfor er det nødvendigt, at holde *RESET** og *HALT** logisk høje for at undgå utilsigtede reaktioner i mikrodatamaten. Dette gøres vha. en timerkreds LM555 i koblingen i figur G.3.



Figur G.3: Power on Reset kredsløbet

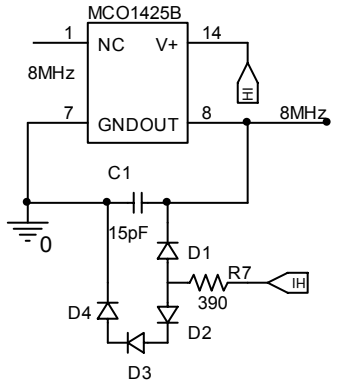
Når timerkoblingen forsynes med strøm vil udgangen på denne (pin 3) gå høj i en hvis periode. Denne periode (t) er bestemt af henholdsvis R1 og C1 jf. ligning G.1. Der er valgt en længere periode end nødvendigt for at være sikker.

$$t = 1{,}1 \cdot R1 \cdot C1 \Rightarrow t = 1{,}1 \cdot 1M\Omega \cdot 0{,}1\mu F = 0{,}11s \qquad (G.1)$$

Kravet er, at *RESET** og *HALT** skal holdes logisk høje i minimum 100ms, hvilket er opfyldt med de valgte komponentværdier. Idet udgangen fra timerkredsen er inverteret i forhold til *RESET** og *HALT** indskydes en inverter mellem timerkredsen og de to indgange. Der er på trigger-indgangen indsat en kontakt til anvendelse ved manuel reset af CPU'en.

G.1.4 Clock

Det er valgt at benytte en clockfrekvens på 8MHz til CPU'en, da tidligere erfaringer viser, at dette er tilstrækkeligt til at udføre de ønskede operationer. Signalet skal være et almindeligt TTL-signal, hvorfor der benyttes en krystaloscillator koblet op jf. diagrammet i figur G.4.



Figur G.4: Kredsløb for 8MHz clockgenerator til CPU'en

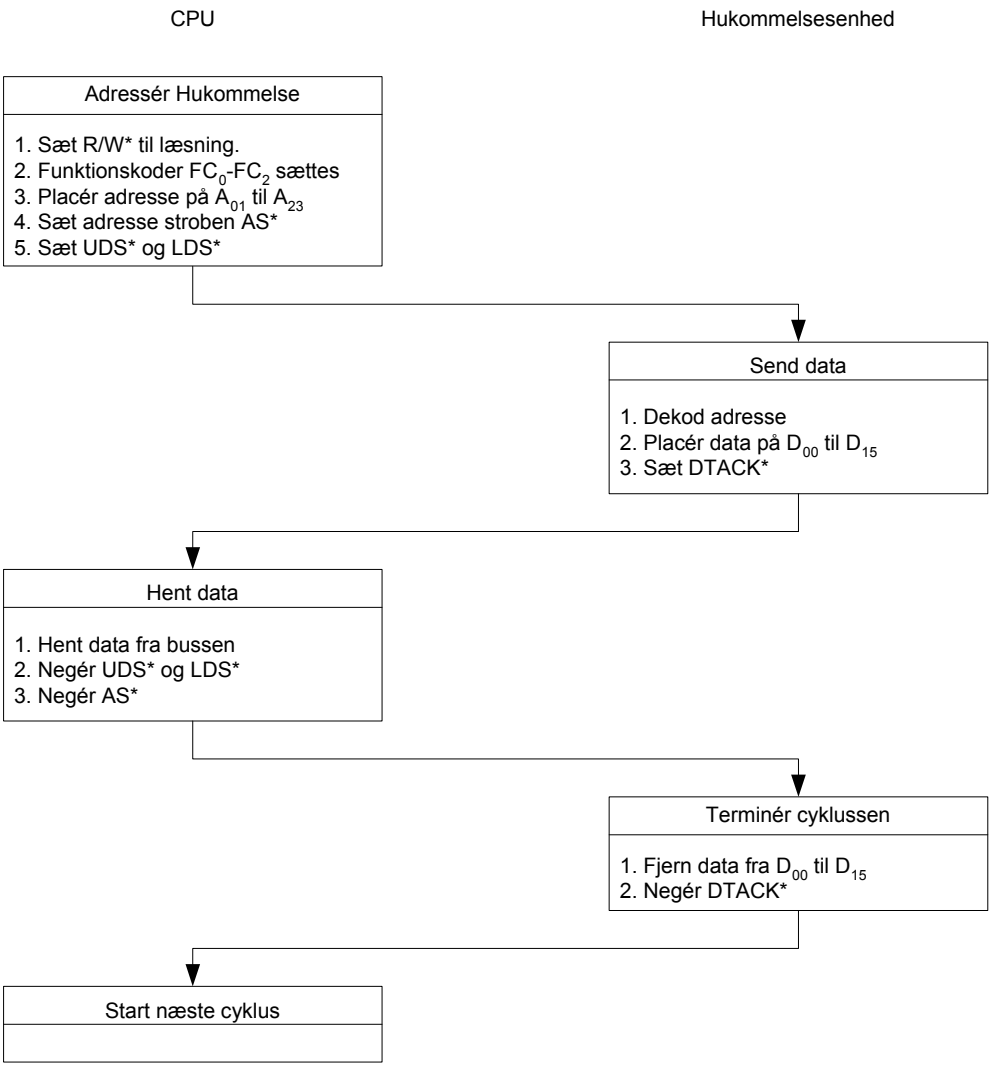
De eksterne dioder, modstanden og kondensatoren er indsat for at dæmpe støjen fra de digitale signaler.
Idet clocksignalet styrer al intern timing i CPU'en er det vigtigt, at denne er meget stabil. Den anvendte krystaloscillator har en nøjagtighed på $\pm 100\,ppm$, hvilket svarer til en frekvens på mellem $7,9992MHz$ og $8,0008MHz$.

G.2 Læse/skrive cyklus timing

Timingen i mikrodatamatsystemet er en vigtig faktor. Hvis CPU'en sættes op til at tilgå hukommelse fra et langsomt RAM/ROM-lager, kan det resultere i en flaskehals i mikrodatamatsystemet.
Timingen i mikrodatamatsystemet fastsættes af CPU'ens clock-generator.
Clock-signalets periode-tid betegnes en clockcyklus, og det er samtidig systemets tidsreference. Alt der sker i systemet, styres tidsmæssigt af dette signal. Systemets læse- og skrivecykluser varer minimum 4 clock-cykluser.

G.2.1 Læsecyklus

Følgende figur viser kommunikationen, ved en læse-cyklus, mellem en bus master og en bus slave. I dette tilfælde mellem CPU'en og hukommelsen.



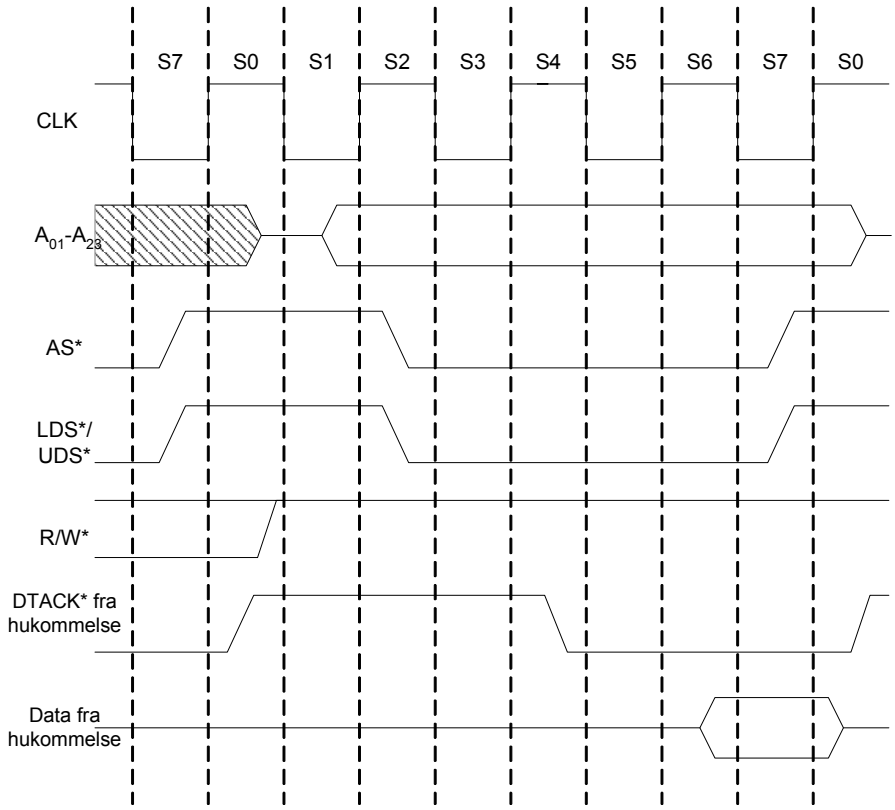
Figur G.5: Flow diagram af en læse cyklus

Højre side viser bus-slavens (hukommelsens) handlinger, mens venstre side viser bus-masterens (CPU'en).
I det følgende gennemgås flowet af en læsecyklus. Læsecyklussen starter ved at sætte en adresse på adressebussen. Derefter sættes R/W* høj, idet der er tale om en læsecyklus. AS*, LDS* og UDS* sættes, hvilket gennem kontrollogikken indikerer hvilken af hukommelseskredsene, der skal adresseres til. Funktionskoderne (FC0 til FC2) sættes ligeledes, men anvendes kun til generering af VPA-signal, der anvendes til vektoriseret interrupt.
Når slaven har modtaget og dekodet adressen på adresse bussen, sættes de relevante data på databussen. Slaven sætter DTACK*, så CPU'en informeres om, at der er gyldige data på databussen, og at den kan fortsætte med processen.
CPU'en læser nu data ind fra databussen, og negerer herefter AS*, LDS* og UDS*. Når disse negeres, fjerner slaven den data der var blevet sendt ud på databussen, og negerer DTACK*. Hermed er en læsecyklus gennemført, og systemet kan gå videre til næste cyklus.

Timing diagram for en læsecyklus

En buscyklus består af minimum 4 clock perioder, hvor hver periode kan inddeles i 2 states, én for høj og én for lav. Så en buscyklus består af minimum 8 states (S0-S7).

Figur G.5 afspejler ikke tiden det tager for cyklussen at blive gennemført. Til det skal der anvendes et timing diagram.



Figur G.6: Læse cyklus for 68000 CPU'en.

Figur G.6 viser et timingdiagram for 68000 CPU'en, hvor hver clockcyklus repræsenteres af S0-S7.

S0 – Alle signaler med undtagelse af R/W* er lave. R/W* er høj for at indikere at der er tale om en læse cyklus.

S1 – Adressebussen får gyldig data.

S2 – AS* skifter til lav, så der indikeres at adressen på bussen er gyldig. LDS* og UDS* går lav for at indikere tilgåelse af hukommelse.

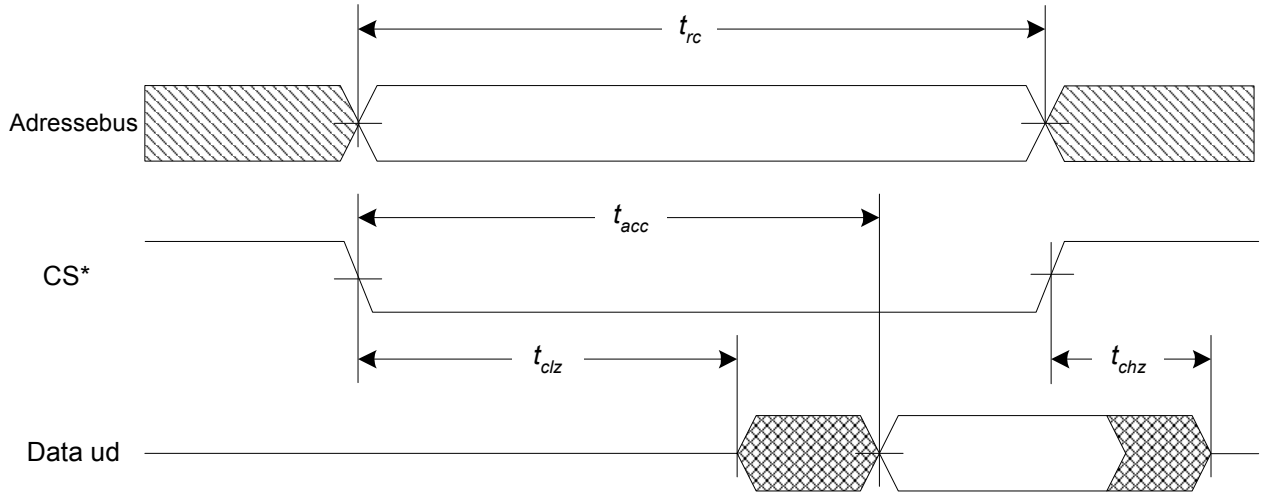
S4 – Hukommelsen reagerer på tilgangen ved at sætte DTACK*.

S6 – Data sættes på databussen af hukommelsen.

S7 – AS*, LDS* og UDS* negeres igen efter CPU'en har læst data fra bussen, som reaktion på dette negeres DTACK og dataene fjernes fra bussen.

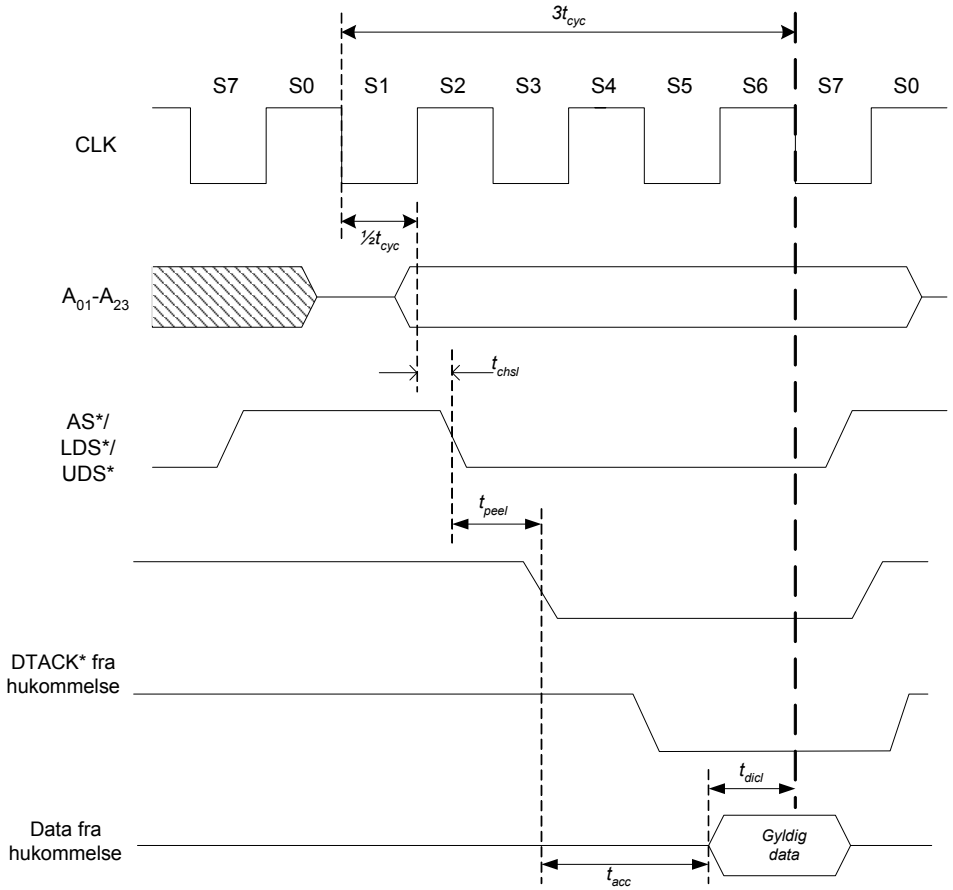
Efter endt læsecyklus kan der påbegyndes en ny proces.

Figur G.7 viser timing diagrammet for en hukommelsesenhed, og illustrerer kommunikationen med mikrodatamaten.



Figur G.7: Læse cyklus for hukommelsesenhed

Læsecyklus-tiden t_{rc}, er det tidsrum hvor adressen er gyldig. CS* bliver sat, og data sættes på data bussen til tiden t_{clz} efter CS* blev sat. Dataene på bussen erklæres dog først gyldige, når tilgangstiden t_{acc} er passeret. CPU'en kan ikke læse dataene før t_{acc} er overskredet. Når CS* negeres til høj går der tiden t_{chz} til dataene er fjernet fra databussen.
Der er en del timingskrav, der skal overholdes når hukommelsesenheden sættes sammen med CPU'en. Hukommelsesenhedens tilgangstid t_{acc}, skal være tilstrækkelig i forhold til CPU'ens data setup tid når den læser t_{dicl}. Af figur G.8 fremgår det, at hvis t_{acc} er for lang, vil det bevirke at t_{dicl} bliver forlænget, og hvis det sker, skal der indsættes wait-states. Disse vil blive indsat efter S4, indtil DTACK skifter til lav.



Figur G.8: Læse cyklus diagram for CPU og hukommelsesenhed.

På figur G.8 ses en tid på 3 clockperioder, $3t_{cyc}$. Perioden starter med S1 og slutter med S6, dette er tiden fra S0 til CPU'en har læst fra hukommelsen.
Tidsperioden starter med en halv periode ved S1, $½t_{cyc}$, til adressen er klar. Derefter går der tid fra S1 til AS*, LDS* og UDS* går lave, denne tid hedder t_{chsl}. Når AS*, LDS* og UDS* bliver lave, bliver CS* på hukommelsen det også efter tiden t_{peel}, denne tid er bestemt af forsinkelsen i PEEL-kredsen der anvendes til kontrollogik.
Når hukommelsen får CS*, bevirker det at en påbegyndelse af tilgang af data går i gang, denne tid er t_{acc}. Efter hukommelsestilgangstiden t_{acc} kommer data setup tiden t_{dicl}.
Denne proces skulle som tidligere nævnt være afviklet indenfor tre clockcyklusser for at undgå wait-states. Der kan derfor opstilles en ulighed:

$$\begin{aligned} &½t_{cyc} + t_{chsl} + t_{peel} + t_{acc} + t_{dicl} \le 3t_{cyc} \Leftrightarrow \\ &t_{acc} \le 3t_{cyc} - ½t_{cyc} - t_{chsl} - t_{peel} - t_{dicl} \end{aligned} \qquad (G.2)$$

Tilgangstiden t_{acc} for hukommelsen skal beregnes, så der kan vælges en passende hukommelsesenhed til CPU'en. Følgende tider er taget fra databladet for M68000.

Parameter	Symbol	M68K ved 8 Mhz		Enhed
		Min	Max	
Cycle time	t_{cyc}	125	250	*ns*
Clock high to AS* asserted	t_{chsl}	3	60	*ns*
D-in valid to clock low (setup time on read)	t_{dicl}	10	-	*ns*

Tabel G.2: Kilde Motorola M68000 afsnit 3

Der anvendes en PEEL til adressedekodning, derfor er PEEL'ens tid, t_{peel}, fra den har input til den genererer output, vigtig. PEEL'ens datablad viser at tiden er 15 *ns*.

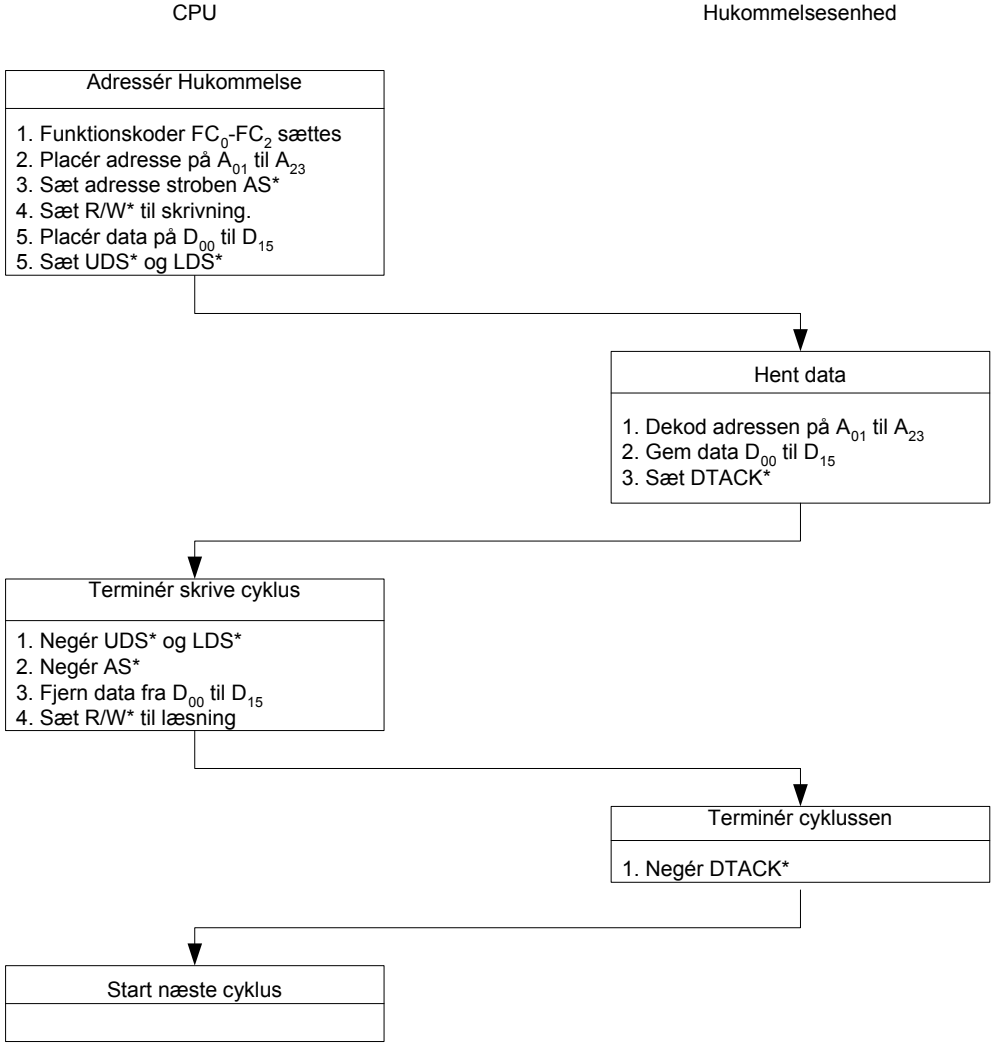
Indsættes værdierne i uligheden fås:

$$3 \cdot 125 - ½ \cdot 125 - 60 - 15 - 10 \geq t_{acc} \Rightarrow t_{acc} \leq 227,5ns \qquad (G.3)$$

Dette betyder ifølge ligning G.3, at hukommelsesenheden skal have en tilgangstid, der er mindre end 227,5*ns*, hvis der skal undgås wait-states. Dette krav til hukommelse gælder for både RAM og ROM.

G.2.2 Skrive cyklus

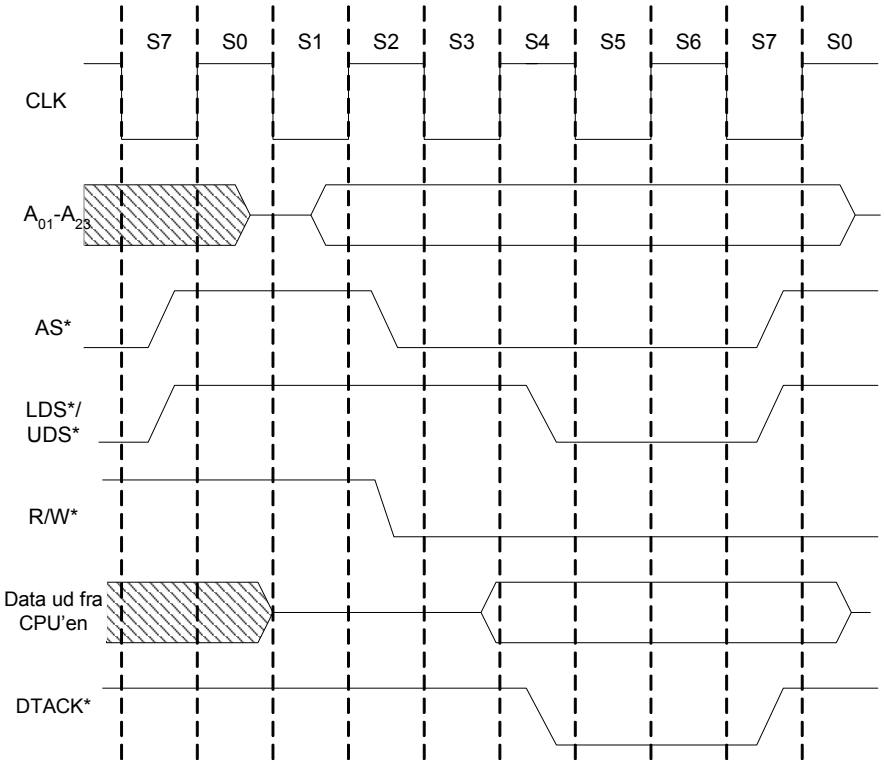
Figur G.9 viser flow-diagrammet for en skrivecyklus. Her er dataenes vej dog modsat, fra CPU'en til hukommelsen.



Figur G.9: Flow diagram af en skrive cyklus

Timing diagram for skrive cyklus

Figur G.10 viser timing diagram for en skrive cyklus.



Figur G.10: Timing diagram for en skrive cyklus

S1 – Adressen er gyldig på bussen.

S2 – AS* bliver sat lav af CPU'en for at indikere, at der nu er gyldig data på adressebussen. Samtidig sættes R/W* lav for at informere hukommelsen om, at der er tale om en skrivecyklus.

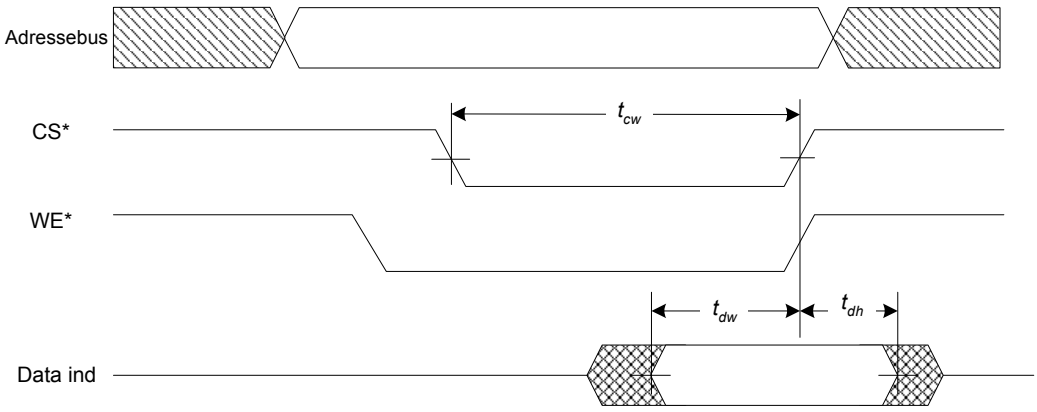
S3 – Gyldig data placeres på databussen.

S4 – Datastroberne LDS* og UDS* sættes lave for sammen med AS*, at fortælle hvilken adresse, der skal skrives til. DTACK* skifter lav for at indikere, at der nu er gyldig data på databussen.

S5–S6 Data skrives i hukommelse.

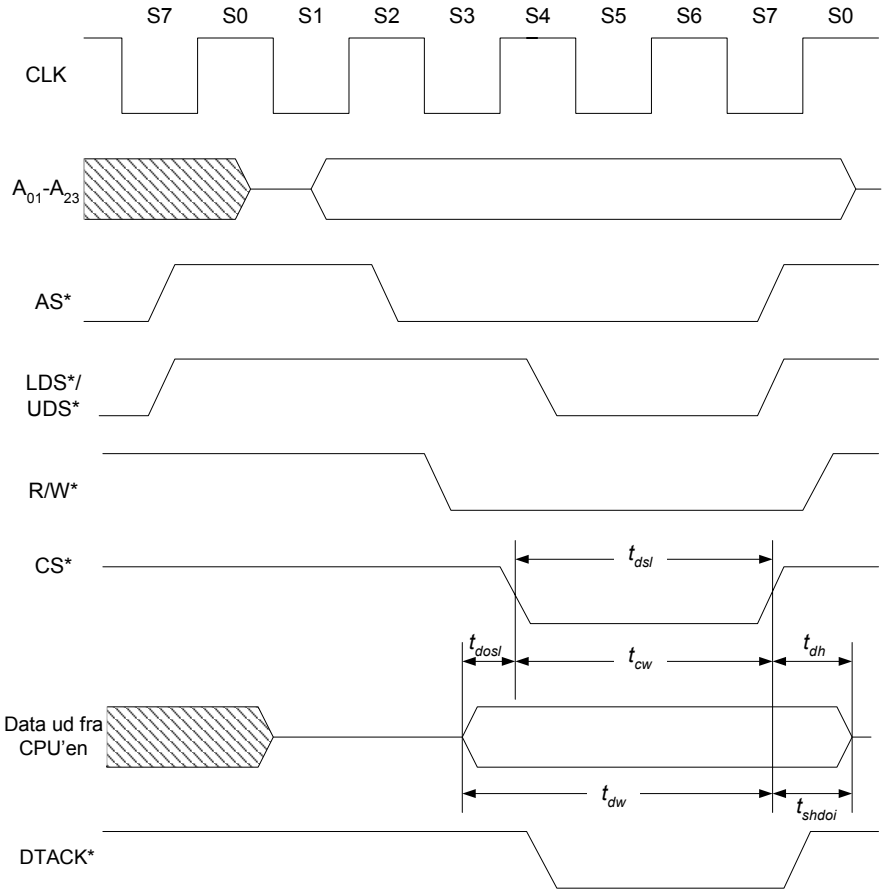
S7 – AS*, UDS* og LDS* negeres som resultat af, at DTACK* skiftede lav i tilstand S4. DTACK* negeres også selv i tilstand S7.

I tilstand S0 er indholdet af adresse- og databussen ugyldigt, så en ny instruktion kan påbegyndes.



Figur G.11: Skrive cyklus timing diagram for hukommelsen

Figur G.11 viser et skrive cyklus timing diagram for hukommelsen. Skrivecyklussen er styret af CS*, som delvist er styret af UDS*/LDS*(jf. afsnit G.3). WE* er tilsluttet R/W* på CPU'en. Tiderne for hukommelsesenheden, der er angivet på figuren, skal overholdes af CPU'en, der sættes sammen med enheden for at disse kan samarbejde.
Skrivecyklussen starter, når CS* sættes lav. Når CS* sættes høj, indikerer det at skrivecyklussen er færdig. Denne tid benævnes t_{cw}, og er den tid, enheden minimum skal have for at kunne læse data ind i hukommelsen. Tiden t_{dw} er tiden det tager, fra der er gyldig data på databussen til det skrives ind i hukommelsesenheden, når CS* går høj. Tiden t_{dh} er udtryk for, hvor længe dataene holdes på databussen, til de erklæres ugyldige.



Figur G.12: Samlet timing for hukommelse og CPU

Timing diagrammet på figur G.12 er en sammensætning af hukommelsesenhed og CPU.
Tiden t_{dsl} fra CS* går lav til den igen går høj, er for CPU'en, denne tid svarer til t_{cw} for hukommelsesenheden. CPU'ens tid t_{dsl} skal derfor være større end hukommelsens t_{cw}, ellers vil der blive indsat wait-states.
Derfor opstilles denne ulighed:

$$t_{dsl} > t_{cw} \qquad (G.4)$$

Følgende værdier aflæses fra databog for Motorola 68000.

Parameter	Symbol	M68K ved 8 Mhz		Enhed
		Min	Max	
DS width asserted	t_{dsl}	140	-	*ns*
Clock high to AS* asserted	t_{dosl}	40	-	*ns*
D-in valid to clock low (setup time on read)	t_{shdoi}	40	-	*ns*

Tabel G.3: Kilde Motorola M68000 afsnit 3

Ved indsættelse af t_{dsl} fås:

$$140ns > t_{cw} \qquad (G.5)$$

Tiden t_{cw} for hukommelsesenheden skal dermed være mindre end 140*ns*. Input data setup tiden t_{dw} for hukommelsesenheden skal være mindre end den tid, der er gyldige data på databussen, altså før LDS*/UDS* går lav.

Dermed opstilles følgende ulighed:

$$t_{dsl} + t_{dosl} > t_{dw} \qquad (G.6)$$

Og der fås ved indsættelse at:

$$140ns + 40ns > t_{dw} \Rightarrow 180ns > t_{dw} \qquad (G.7)$$

Input data setup tiden t_{dw} for hukommelsesenheden, skal dermed være mindre end 180*ns*. Efter CS* er gået høj skal hukommelsesenheden holde dataene i tiden t_{dh}, og denne tid skal være mindre end den tid CPU'en holder dataene gyldige på databussen, efter CS* er negeret.
Så fås denne ulighed:

$$t_{shdoi} > t_{dh} \qquad (G.8)$$

Tiden t_{shdoi} indsættes:

$$40\ ns > t_{dh} \qquad (G.9)$$

Hukommelsesenhedens input data hold tid t_{dh} skal dermed være mindre end 40 *ns*.

De fundne tider skal bruges til bestemmelse af hukommelsesenhed.

Tidskravene er som følger:

$$227,5ns \geq t_{acc}$$
$$180\ ns > t_{dw}$$
$$40\ ns > t_{dh}$$

Det er fundet at hukommelsesenheden SRM20100LC$_{70}$ lever op til disse krav:

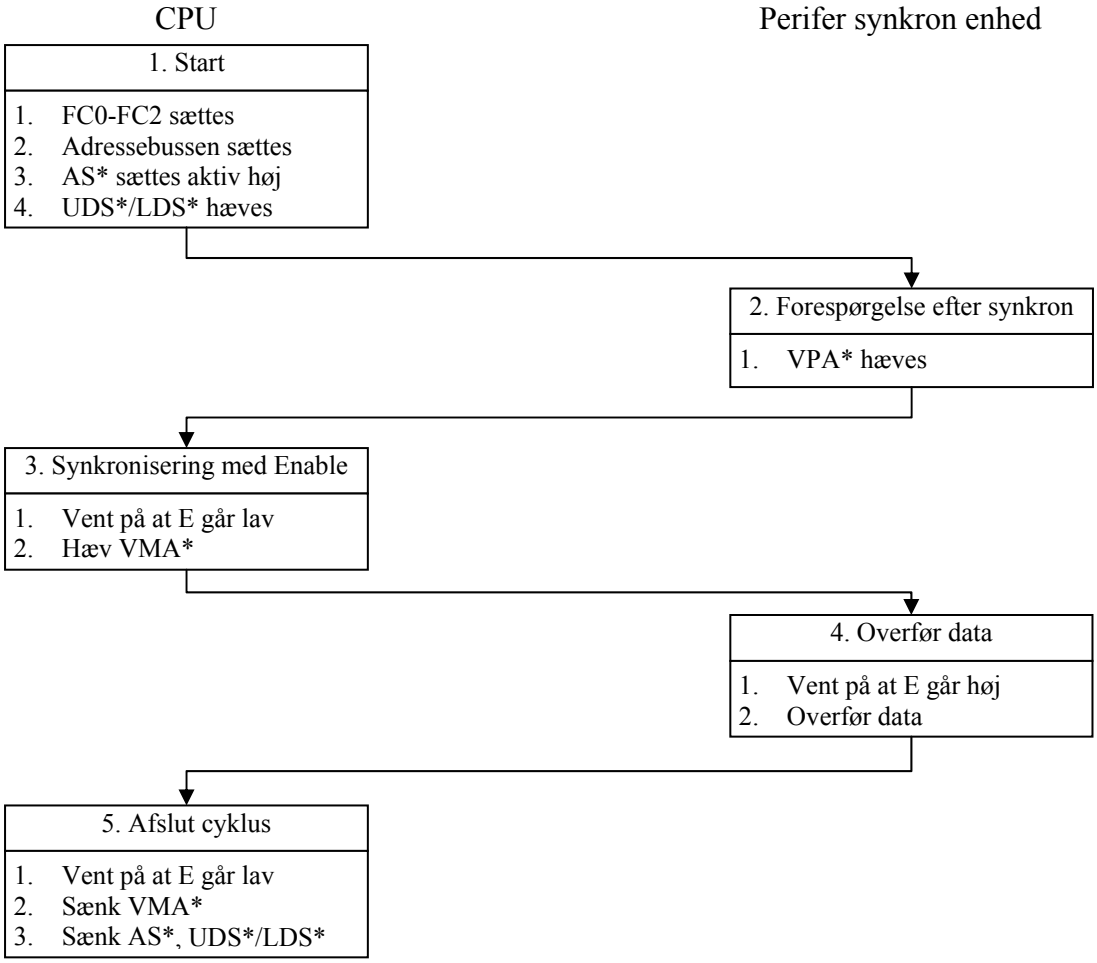
Parameter	Symbol	SRM20100LC$_{70}$		Enhed
		Min	Max	
Address access time	t_{acc}	-	70	*ns*
Input data setup time	t_{dw}	30	-	*ns*
Input data hold time	t_{dh}	-	30	*ns*

Tabel G.4: Tider fra datablad for hukommelsesenhed

RAM-enheden er på 1Mbit (128 kb), hvilket lever op til kravet omkring størrelsen på enheden.
Som ROM-enhed anvendes en AM29F010B, der med en worst case tilgangstid på 120ns, også lever op til CPU'ens krav.

G.3 Kontrollogik

G.3.1 Synkron buskontrol



Figur G.13: Flow diagram for synkron buskontrol

På figur G.13 ses et flow diagram for en synkron buscyklus, hvilket vil blive beskrevet efterfølgende:

1. Som det første vil funktionsudgangene på CPU'en blive sat alt efter, hvilken tilstand befinder sig i. Efterfølgende lægges den ønskede adresse på adressebussen og AS* går høj. Afhængigt af om det er det øvre eller nedre adresseområde der tilgås, vil henholdsvis UDS/ eller LDS* sættes høj.
2. Den perifere enhed hæver VPA* for at igangsætte en synkron buscyklus.
3. CPU'en synkroniseres med enheden ved at vente på at E-clocken går lav. Dernæst hæves VMA* for at informere den perifere enhed om, at en synkron buscyklus er iværksat.
4. Den eksterne enhed venter nu på at E går høj, igen af hensyn til synkronisering. Derefter overføres dataene til CPU'en.
5. CPU'en venter nu på, at E går lav, hvorefter den sænker VMA* efterfulgt af AS* samt UDS*/LDS*.

G.3.2 Kode til PEEL-kreds (adressedekoder)

Syntaksen for koden i PEEL-kredsen er som følge:
!: NOT
&: AND
+: OR
.COM: Angiver at det er en kombinatorisk udgang.
.OE: Aktiverer udgangen.

```
"Adressedekoder:
CS_ROM.COM = !(A18 + A19 + A20 + A23) & !AS;     "ROM aktiveres, hvis hverken RAM (A18),
CS_ROM.OE = 1;                                   "ADC (A19) eller ACIA (A23) er aktiveret

CS_RAM.COM = A18 & !A19 & !A20 & !A23 & !AS;     "RAM aktiveres, hvis A18 og kun A18 er
CS_RAM.OE = 1;                                   "høj.

CS_ACIA.COM = !A18 & !A19 & !A20 & A23 & !AS & !VMA;        "ACIA aktiveres, hvis A23 og
CS_ACIA.OE = 1;                                             "VMA aktiveres.

CS_BUF.COM = !A18 & A19 & !A20 & !A23 & !AS;     "A/D-konverter aktiveres hvis A19
CS_BUF.OE = 1;                                   "aktiveres.

CS_KAL.COM = !A18 & !A19 & A20 & !A23 & !AS;     "Chipselect for digitale modstande.
CS_KAL.OE = 1;

"DTACK er en data acknowledge til M68k, der indikerer, at data er klar på
"databussen. Alle kredse på databussen er tilstrækkeligt hurtige, så DTACK
"kan sættes direkte efter chipselect for de enkelte kredse.
DTACK.COM = !CS_ROM + !CS_RAM + !CS_ACIA + !CS_BUF + !CS_KAL;
DTACK.OE = 1;

"VPA er et signal, der fortæller, at CPU'en skal køre med synkron databus. Bruges til
"ACIA. Bruges også til autovektoriserede interrupts.
VPA.COM = !A18 & !A19 & !A20 & A23 & !AS;
VPA.OE = 1;
```

G.3.3 Interruptcyklus

I tabel G.5 ses sekvensen af en interruptcyklus:

Interrupt
1. Igangværende instruktion udføres. 2. Programtælleren gemmes. 3. CPU'ens statusregister gemmes. 4. Der hoppes til den pågældende inturruptrutine. 5. CPU'ens statusregister gendannes. 6. Der hoppes til den gemte programtæller.

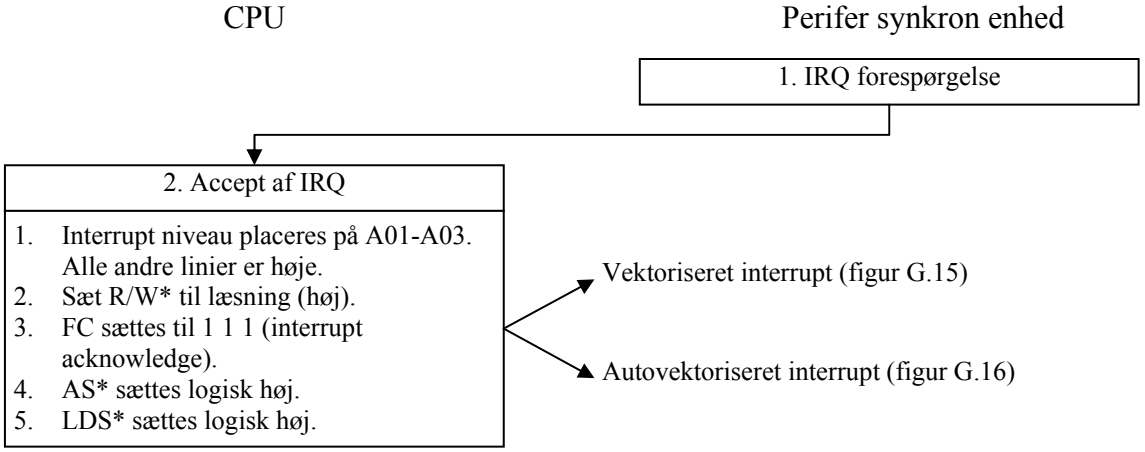
Tabel G.5: Interruptcyklus

1. Den igangværende instruktion udføres, da den anvendte processor ikke er i stand til at standse midt i en instruktion og senere genoptage denne.
2. Indholdet af programtælleren gemmes, således at CPU'en efter endt interruptudførelse er i stand til at genoptage det igangværende program.

3. Indholdet af statusregistret gemmes, så CPU'en kan genoptage udførelsen af programmet i samme tilstand som den forlod det.
4. Der hoppes til det register, der indeholder starten af den pågældende interruptrutine og denne udføres.
5. CPU'ens statusregister gendannes til tidspunktet lige inden udførelse af interruptet.
6. CPU'en hopper til det sted, den var kommet til inden interruptet.

G.3.4 Interrupt acknowledge

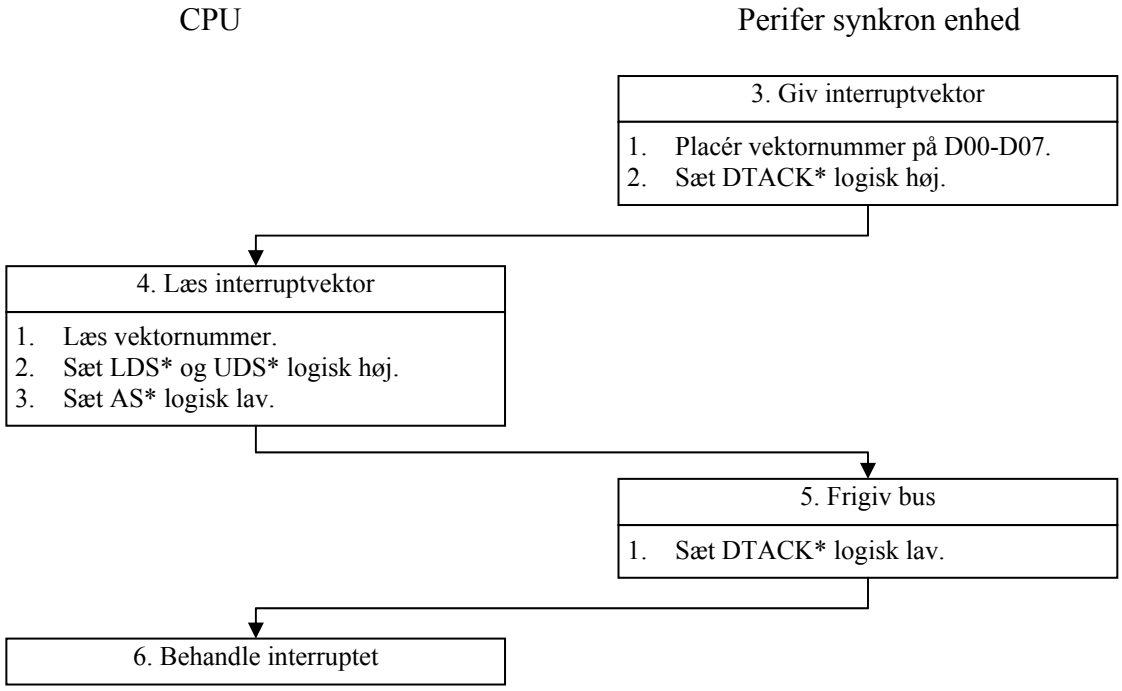
Interrupt acknowledge vil ske uanset typen af interrupt.



Figur G.14: Generelt interrupt acknowledge

1. En perifer enhed forespørger et interrupt.
2. Det pågældende interruptniveau placeres på adressebussens tre første bit (A01-A03). R/W* sættes til læsning, hvorefter det på funktionsbenene kan aflæses, om CPU'en giver en bekræftelse på interruptet. Efterfølgende sættes AS* samt LDS* logisk høj. Afhængigt af typen af interrupt udføres forskellige operationer, hvilket fremgår af figur G.15 og G.16.

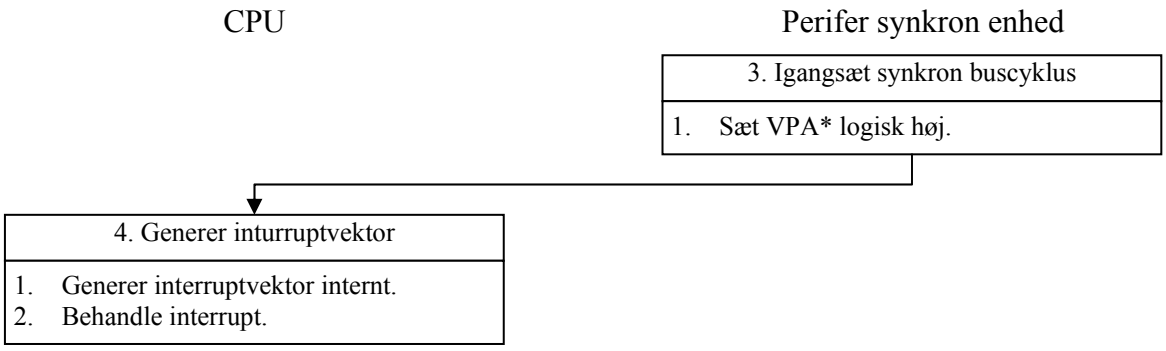
G.3.5 Vektoriseret interrupt



Figur G.15: Flow chart for vektoriseret interrupt

3. Interruptvektoren placeres på databussens første 8 bit, hvorefter DTACK* går logisk høj.
4. CPU'en læser vektornummeret, sætter LDS*, UDS* logisk høj samt AS* logisk lav.
5. Bussen frigives igen ved at sætte DTACK* logisk lav.
6. CPU'en behandler interruptet.

G.3.6 Autovektoriseret interrupt



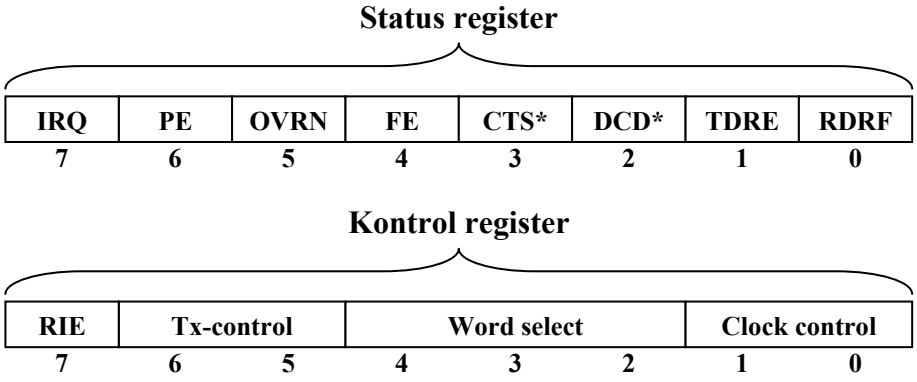
Figur G.16: Flow chart for autovektoriseret interrupt

3. Den perifere enhed sætter VPA* logisk høj for at starte en synkron buscyklus.
4. CPU'en genererer interruptvektoren internt, hvorefter interruptet behandles.

G.4 Perifere enheder

G.4.1 Registre i ACIA

ACIA 6850 indeholder fire registre: Sende-, modtage-, status- samt kontrolregister, hvoraf de to sidste fremgår af figur G.17.



Figur G.17: Opbygningen af status- og konrolregister i ACIA 6850

Status register

Dette register kan kun læses og indeholder forskellige oplysninger om ACIA'ens tilstand, hvilket fremgår af nedenstående punktopstilling.

- **IRQ (7):** Angiver om ACIA'en ønsker af interrupte CPU'en.
- **PE – Parity Error (6):** Angiver om der er paritetsfejl.
- **OVRN – Reciever Overrun (5):** Angiver at ACIA'en har modtaget en ny byte inden den foregående er blevet læst.
- **FE – Framing Error (4):** Bliver sat hvis der mangler et stopbit.
- **CTS* - Clear To Send (3):** Afspejler tilstanden af CTS* indgangen på ACIA'en.
- **DCD* - Data Carrier Detect (2):** Viser om der er fejl i indgående data.
- **TDRE – Transmitter Data Register Ready (1):** Angiver at senderegistret er tomt og er klar til at modtage nye data.
- **RDRF – Reciever Data Register Full (0):** Angiver at modtageregistret er fuldt

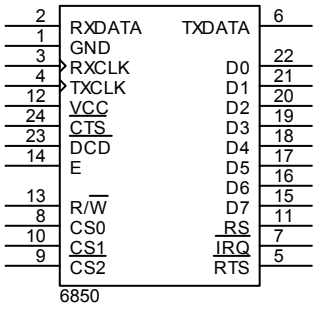
Kontrol register

Register der anvendes til konfigurering af ACIA'en, herunder overførelseshastigheder samt opbygning af data (antal databit, paritet o.l.). Beskrivelse af de enkelte indstillinger i registret findes nedenfor:

- **RIE – Reciever Input Enable (7):** Angiver om ACIA'en skal interrupte ved modtaget data.
- **Tx-control (6-5):** Angiver om ACIA'en skal interrupte ved sendt data.
- **Word select (4-2):** Angiver antallet af databits, stop bits samt om der skal anvendes paritetsbit.
- **Clock control (1-0):** Angiver hvor meget indgangsclocken til ACIA'en skal divideres med.

G.4.2 Benbeskrivelser for ACIA'en

Den anvendte ACIA er af typen MC6850 og har 24 ben i DIP jf. nedenstående figur.



Figur G.18: ACIA 6850

#	Ben (antal)	Beskrivelse
	D(0-7)	Databus
2	**RXDATA**	Indgang til seriel modtagelse af data.
3	**RXCLK**	Clock til seriel modtagelse af data.
4	**TXCLK**	Clock til seriel afsending af data.
5	**RTS**	ACIA'ens anmodning om at sende data. Anvendes ikke.
6	**TXDATA**	Udgang til seriel afsending af data.
7	**IRQ***	Udgang der angiver om ACIA'en ønsker at interrupte CPU'en.
8	**CS0**	Anvendes ikke.
9	**CS2***	Chip Select af ACIA.
10	**CS1**	Anvendes ikke.
11	**RS**	Anvendes til selektering af de interne registre.
12	**VCC**	+5V forsyningsspænding.
13	**R/W***	Anvendes til selektering af de interne registre.
14	**E**	Clock til synkroniseringen med CPU'en.
23	**CTS***	Angiver om ACIA'en har tilladelse til at afsende data. Dette kan ligeledes aflæses i statusregistret.
24	**DCD**	Angiver om der er fejl i indgående data.

Tabel G.6: Beskrivelser af benene på ACIA 6850

H PC software

```
/*****************************************************************************
   Fil         : PC.cpp (kun til Windows)
   Forfattere  : Gr 413 2002
   Dato        : 290402-290502
*****************************************************************************/
/* Inkluderede filer */
#include <stdio.h>   // Bruges af printf()
#include <conio.h>   // Bruges af getch()
#include <vcl.h>     // Krævet af winbase.h
#include <winbase.h> // Bruges af diverse kommunikationsfunktioner

   /* Globale variable */
   HANDLE ComPort = NULL;                    // Handle til håndtering af COM-porten
   DCB dcbCommPort;                          // Indeholder instillingerne for COM-porten
   const char FilNavn[50] = "PROGRAM.REC" ;  // Angiver navnet på datafilen
   const char ComNummer[30] = "COM1" ;       // Angiver navnet på den anvendte COM-port
   char ModtagData[1];                       // Indeholder modtaget data

   /* Erklærede funktioner */
   void Upload(void) ;                    // Upload-menuen
   void Kalibrering(void) ;               // Kalibrerings-menuen
   void KalKanal(char) ;                  // Funktion til kalibrering af kanaler
   void Taerskel (void) ;                 // Funktion til indstilling af tærskelværdien
   void Opstart(void) ;                   // Opstarts-menuen
   void Afslut(void) ;                    // Afslutnings-menuen
   void menu(void) ;                      // Visning af hovedmenu
   void ModtagChar(void);                 // Modtager en karakter fra COM-porten
   void Initialisering(void) ;            // Sætter COM-porten op
   void SendModtagChar(char) ;            // Sender en karakter og afventer svar
   void SendFil(const char filnavn[50]) ; // Sender datafilen til mikrodatamaten
/*****************************************************************************
                                 Main
*****************************************************************************/
int main(void)
{
    int status = 0 ;

    Initialisering();   // Opsætning af COM-porten for at modtage svar fra TS2-monitor
    Initialisering();   // Opsætning af COM-porten til anden kommunikation
    menu();             // Start af menu

    return status ;
}
/*****************************************************************************
                                Hovedmenu
Viser hovedmenuen
*****************************************************************************/
void menu(void)
{
    char valg ;
    if (!SetCommState (ComPort, &dcbCommPort)) // Undersøger om porten kan åbnes
    {
        printf(" %s-porten kunne ikke aabnes!!!\n\nTryk en tast for at \
lukke programmet...", ComNummer) ;
        getch() ;
    }
    else
    {
        while(valg != 27)  // Kører indtil der trykkes på knappen Esc (ASCII tegn 27)
        {
```

```
            clrscr();
            printf("**********   HOVEDMENU   **********\n\n\n");
            printf(" 1   : Upload mikrodatamatprogram\n") ;
            printf(" 2   : Kalibrering\n") ;
            printf(" 3   : Start af museprogram og afslut\n") ;
            printf(" Esc : Afslut menu\n") ;
            valg = getch() ;                      // Gemmer det indtastede tegn i "valg"
            switch(valg)                          // Starter den case der er blevet valgt
            {
                case '1' : Upload() ;         // Undermenuen for upload af program
                  break ;                     // Afslutning af casen upload
                case '2' : Kalibrering() ;    // Undermenuen for kalibrering af kanaler
                  break ;                     // Afslutning af casen kalibrering
                case '3' : Opstart() ;        // Undermenuen for start af museprogram
                  break ;                     // Afslutning af casen opstart
                case 27 : Afslut() ;          // Undermenuen for afslutning af program
                  break ;                     // Afslutning af casen afslut
                default : printf("\n\n\n Forkert kommando !!!\n\n\n");
                Sleep(1000);                  // Skriver teksten i 1 sekund
                  break ;                     // Afslutning af default
            } // End for switch
        } // End for while
    } // End for else
} // End for menu(void)
/*****************************************************************************
                                Upload
Menu til anvendelse ved upload af program til mikrodatamat.
*****************************************************************************/
void Upload(void)
{
    char upvalg ;
    while(upvalg != 27)     // Kører indtil der trykkes på knappen Esc (ASCII tegn 27)
    {
        clrscr();
        printf("\n\n\n j   : Begynd upload \n Esc : Hovedmenu \n\n\n") ;
        upvalg = getch() ;               // Gemmer det indtastede tegn i "upvalg"
        switch (upvalg)                  // Starter den case der er blevet valgt
        {
            case 'j': SendFil(FilNavn) ; // Går til undermenuen for send fil
                upvalg = 27 ;            // til mikrocomputeren
            break ;                      // Afslutning af casen sendfil
            case 27: ;                   // Går til hovedmenuen
            break ;                      // Afslutning af case
            default : printf("\n\n\n Forkert kommando !!!\n\n\n");
                Sleep(1000);             // Skriver teksten i 1 sekund
            break ;                      // Afslutning af default
        } // End switch
    } // End while
} // End void Upload
/*****************************************************************************
                              Kalibreringsmenu
Viser kalibreringsmenuen og kalder de relevante kalibreringsfunktioner
*****************************************************************************/
void Kalibrering(void)
{
    char kalvalg ;

    while(kalvalg !=27)
    {
        clrscr();
        printf("\n\n\n 1-4 : Kanal der oenskes kalibreret \n 5   : Kalibrering af \
offset- og taerskelvaerdier \n Esc : Hovedmenu \n\n\n") ;
        kalvalg = getch() ;
        switch(kalvalg)
        {
```

```
            case '1': KalKanal('1') ; // Funktionen 'KalKanal' kaldes med parametren '1'
            break ;
            case '2': KalKanal('2') ;
            break ;
            case '3': KalKanal('3') ;
            break ;
            case '4': KalKanal('4') ;
            break ;
            case '5': Taerskel();     // Funktionen 'Taerskel' kaldes
            break ;
            case 27: ;
            break ;
            default : printf("\n\n\n Forkert kommando !!!\n\n\n") ;
                Sleep(1000) ;
            break ;
        } // End switch
    } // End while
} // End void Kalibrering
/*****************************************************************************
                        Kalibrering af de fire kanaler
Funktion til anvendelse ved kalibrering af kanalerne
*****************************************************************************/
void KalKanal(char kanal)
{
    char start ;
    while(start != 27 )
    {
        clrscr() ;
        printf("\n\n\n Du har valgt at kalibrere kanal %d\n", (kanal-48)) ;
                                                    // -48 af hensyn til ASCII-værdi
        switch(kanal)
        {
            case '1': printf(" ( Bevaegelse mod hoejre ) \n\n\n") ;
            break;
            case '2': printf(" ( Bevaegelse mod venstre ) \n\n\n") ;
            break;
            case '3': printf(" ( Bevaegelse op ) \n\n\n") ;
            break;
            case '4': printf(" ( Bevaegelse ned ) \n\n\n") ;
            break;
        }

        printf(" Naar start vaelges, spaend da musklen der bruges til kanal %d i 5 sek.",
            (kanal-48)) ;
        printf("\n\n\n s  : Start \n Esc: Tilbage \n\n\n");
        start = getch() ;

        switch(start)
        {
            case 's':
            printf(" Mikrodatamaten kontaktes...") ;
            SendModtagChar(kanal) ; // Sender karakteren indeholdt i 'kanal' og
            clrscr() ;                                  // modtager bekræftelse
            if (ModtagData[0] == '7')// Undersøger om den korrekte bekræftelse modtages
            {
                printf("\n Kalibreringen er paabegyndt !!!") ;
                ModtagChar() ;
                if (ModtagData[0] == '7')
                {
                    printf("\n Kalibreringen er faerdig !!!\n") ;
                    Sleep(2000) ;
                    start = 27 ;
                }
                else
                {
```

```
                    printf("\n Kalibreringen kunne ikke gennemfoeres!!! Prøv igen...\n");
                    Sleep(3000) ;
                    start = 27 ;
                }
            }
            else
            {
                printf("\n Der opstod en fejl! (Forkert svar fra mikrodatamaten)\n") ;
                Sleep(2000);
            }
        break ;
        case 27: ;
        break ;
        default :
            printf("\n\n\n Forkert kommando !!!\n\n\n") ;
            Sleep(1000) ;
        break ;
        } // End switch
    } // End while
} // End void kanal
/*****************************************************************************
                        Indstilling af tærskelværdi
*****************************************************************************/
void Taerskel(void)
{
    char start ;
    clrscr() ;
    printf("\n Du har valgt at indstille offset- og taerskelvaerdien for de \
fire kanaler!\n");
    printf("\n Naar start vaelges, slap da af i alle muskler indtil kalibreringen \
er udfoert!\n") ;
    printf("\n\n\n j  : Start \n Esc: Tilbage \n\n\n");
    start = getch();
    switch(start)
    {
        case 'j':
        printf("\n Mikrodatamaten kontaktes...\n") ;
        SendModtagChar('5') ;
        clrscr() ;
        if (ModtagData[0] == '7')
        {
            printf("\n Indstilling af taerskelvaerdi er paabegyndt!\n") ;
            ModtagChar() ;
            if (ModtagData[0] == '7')
            {
                printf("\n Indstilling af taerskelvaerdi er fuldfoert!\n") ;
                Sleep(1000) ;
                start = 27 ;
            }
            else
            {
                printf("\n Kalibreringen kunne ikke gennemfoeres!!! Proev igen...\n") ;
                Sleep(3000) ;
                start = 27 ;
            }
        }
        else
        {
            printf("\n Der opstod en fejl! (Forkert svar fra mikrodatamaten)\n") ;
            Sleep(2000);
        }
        break ;
        case 27: ;
        break ;
        default :
```

```
            printf("\n\n\n Forkert kommando !!!\n\n\n") ;
            Sleep(1000) ;
        break ;
    } // End for switch
} // End for Taerskel
/*****************************************************************************
                                Opstart
Starter museprogrammet og modtager bekræftelse på, at dette er sket.
*****************************************************************************/
void Opstart (void)
{
    char opvalg ;
    while(opvalg != 27)
    {
        clrscr();
        printf("\n\n\n Vil du starte museprogrammet?\n j   : Start program\n \
ESC : Hovedmenu\n\n\n") ;
        opvalg = getch() ;
        switch (opvalg)
        {
            case 'j' :
            printf("\n Museprogrammet startes......\n") ;
            SendModtagChar('6') ;
            if (ModtagData[0] == '7')
            {
                printf("\n Museprogram er startet !!!\n") ;
                printf("\n Programmet afsluttes..... \n") ;
                CloseHandle(ComPort) ; // Porten lukkes
                Sleep(3000) ;
                abort() ;              // Programmet lukkes
            }
            else
            {
                printf("\n Museprogrammet kunne ikke startes !!! \n") ;
                Sleep(2000);
                opvalg = 27 ;
            }
        break ;
        case 27 : ;
        break ;
        default :
            printf("\n\n\n Forkert kommando !!!\n\n\n") ;
            Sleep(1000) ;
        break ;
        } // End switch
    } // End while
} // End void Opstart
/*****************************************************************************
                                Afslut
Viser afslutningsmenu og lukker COM-porten.
*****************************************************************************/
void Afslut (void)
{
    char afslut ;
    void slut(void) ;

    while(afslut !='j')
    {
        clrscr();
        printf("\n\n\n j   : Afslut \n ESC : Hovedmenu \n\n\n") ;
        afslut = getch() ;
        switch (afslut)
        {
            case 'j': CloseHandle(ComPort) ; // COM-porten lukkes
            break ;
```

```
            case 27: menu() ;
                afslut = 'j';
            break ;
            default :
                printf("\n\n\n Forkert kommando !!!\n\n\n") ;
                Sleep(1000) ;
            break ;
        }
    } // End while
} // End void Afslut
/*****************************************************************************
                                OpsaetningCom
Åbner COM-porten og sætter den derefter op
*****************************************************************************/
void Initialisering(void)
{
    void OpenComPort(void);
    void ConfigComPort(void);

    CloseHandle(ComPort);  // Hvis COM-porten i forvejen er åben lukkes den
    OpenComPort();
    ConfigComPort();
}

    void OpenComPort(void)
    {
        ComPort = CreateFile(ComNummer,           // COM-porten åbnes til både...
            GENERIC_READ | GENERIC_WRITE,         // ...læsning og skrivning
            0,
            0,
            OPEN_EXISTING,                        // Porten åbnes kun, hvis den...
            0,                                    // ...eksisterer i forvejen
            0);
    } // End for OpenComPort

    void ConfigComPort(void)
    {
        dcbCommPort.DCBlength = sizeof(DCB);
        GetCommState(ComPort, &dcbCommPort);
        dcbCommPort.BaudRate = 1200;              // Baudrate
        dcbCommPort.fParity = NOPARITY;           // Paritet
        dcbCommPort.ByteSize = 8;                 // Antal databits
        dcbCommPort.StopBits = ONESTOPBIT;        // Antal stopbits
        SetCommState(ComPort, &dcbCommPort);      // Sætter COM-porten op med...
                                                  // ...de nye parametre
        COMMTIMEOUTS CommTimeouts;                // Angiver timeout-parametrene
        CommTimeouts.ReadIntervalTimeout = MAXDWORD;    // Interval mellem bytes
        CommTimeouts.ReadTotalTimeoutMultiplier = MAXDWORD; // -
        CommTimeouts.ReadTotalTimeoutConstant = 8000;   // Samlet læse-timeout i ms
        CommTimeouts.WriteTotalTimeoutMultiplier = 10;  // -
        CommTimeouts.WriteTotalTimeoutConstant = 1000;  // Samlet skrive-timeout i ms
        SetCommTimeouts (ComPort, &CommTimeouts);       // Sætter timeout-parametrene

        if (!SetCommTimeouts (ComPort, &CommTimeouts))
        {
            printf("\n Time-out parametrene kunne ikke saettes for %s-porten!\n\n",
                ComNummer);
        }
    } // End for ConfigComPort
/*****************************************************************************
                                SendModtagChar
Sender en karakter og modtager bekræftelse
*****************************************************************************/
void SendModtagChar(char enkelt)
{
```

```
    TransmitCommChar(ComPort,enkelt); // Sender karakteren indeholdt i 'enkelt'
    ModtagChar();                     // Modtager bekræftelse
}
/*****************************************************************************
                              SendFil
Sender datafilen til mikrodatamaten
*****************************************************************************/
void SendFil(const char filnavn[50])
{
    HANDLE DataFil;
    DWORD FileSize;                   // Indeholder længden på datafilen
    unsigned long BytesWritten = 0;
    unsigned long BytesRead = 0;
    bool WriteError;
    bool ReadError;
    char Buffer[100000];              // Buffer til indholdelse af datafilen

    dcbCommPort.BaudRate = 4800;      // Sætter baudrate til 4800
    SetCommState(ComPort, &dcbCommPort);

    DataFil = CreateFile(filnavn,     // Åbner datafilen til læsning
        GENERIC_READ,
        0,
        0,
        OPEN_EXISTING,
        0,
        0);

    FileSize = GetFileSize(DataFil, NULL); // Finder størrelsen på datafilen
    ReadError = ReadFile(DataFil, Buffer, FileSize, &BytesRead, NULL);//
                                           // Læser datafilen ind i bufferen
    if (ReadError == 0)                    // Undersøger om filen kunne åbnes
        printf(" Filen %s kunne ikke aabnes!\n\n", FilNavn);
    else
    {
        printf("\n Initialiserer mikrodatamaten.......\n");
        WriteError = WriteFile(ComPort, "LOAD", 4, &BytesWritten, NULL);
                    // Sender 'LOAD'-komandoen til TS2-monitoren

        Sleep(1000);
        TransmitCommChar(ComPort,13);  // Sender ASCII-værdien for enter
        printf("\n Data sendes til mikrodatamaten. Vent venligst......\n\n");
        Sleep(1000);

        unsigned int i = 0 ;
        while (i < FileSize)
        {
            if (Buffer[i] == 10)
            {
                Sleep (100);                        // Pauser 100ms ved linieskift
            }
            else
            {
                Sleep (3) ;                         // Pauser 3ms hvis ikke linieskift
                TransmitCommChar(ComPort, Buffer[i]) ;// Sender karakteren på plads i...
            }                                       // ...i bufferen til COM-porten
            i++ ;
        }

        printf("\n %d bytes blev skrevet til mikrodatamaten!\n", FileSize);
        CloseHandle(DataFil); // Lukker datafilen
        Sleep(1000);

        printf("\n Mikrocomputeren startes...\n") ;
        WriteError = WriteFile(ComPort, "JUMP 41000", 10, &BytesWritten, NULL);
```

```
                                                    // Sætter programmet igang
        Sleep(500);
        TransmitCommChar(ComPort, 13); //Sender ASCII-værdien for enter
        printf("\n Mikrocomputeren er klar!\n") ;
    } // End for else

    Sleep(500);

    dcbCommPort.BaudRate = 1200;              // Baudrate sættes tilbage til 1200
    SetCommState(ComPort, &dcbCommPort);

    printf("\n Tryk en tast for at fortsaette......\n");
    getch();
} // End for SendFil
/*****************************************************************************
                                ModtagChar
Sletter først inputbufferen, hvorefter der modtages en karakter
*****************************************************************************/
void ModtagChar(void)
{
    unsigned long event;
    DWORD dwBytesTransferred;

    PurgeComm(ComPort, PURGE_RXCLEAR) ;     // Sletter input-bufferen

    WaitCommEvent (ComPort, &event, NULL) ; // Venter til der modtages en karakter
    if (event = EV_RXCHAR)
    {
        ReadFile (ComPort,
            ModtagData,                     // Variabel til modtaget data
            1,                              // Antal bytes der skal læses
            &dwBytesTransferred,            // Variabel til antal modtaget data
            NULL
            );
    }
} //End ModtagChar
/*****************************************************************************/
```

I Program til mikrodatamaten

I.1 Kildekode

```
*************************************************************
*                                                           *
*              Interrupt tabel                              *
*                                                           *
*************************************************************

               org       $400C8      *Placering af INT1
               bsr       INT1        *Gå til interrupt rutine 1
               rte

               org       $400D0      *Placering af INT2
               bsr       INT2        *Gå til interrupt rutine 2
               rte

               org       $400D8      *Placering af INT3
               bsr       INT2        *Gå til interrupt rutine 2
               rte

               org       $400E0      *Placering af INT4
               bsr       INT4        *Gå til interrupt rutine 4
               rte

               org       $400E8      *Placering af INT5
               bsr       INT4        *Gå til interrupt rutine 4
               rte

               org       $400F0      *Placering af INT6
               bsr       INT4        *Gå til interrupt rutine 4
               rte

*************************************************************
*                                                           *
*              Konstanter                                   *
*                                                           *
*************************************************************

               org       $41000      *Startadresse for program

*ref           equ       $7F         *Default referenceværdi til kalibrering
*refmax        equ       255         *Maxreferenceværdi til kalibrering

acia_kreg      equ       $800001     *Adresse på ACIA'ens kontrol register
acia_sreg      equ       $800001     *Adresse på ACIA'ens status register
acia_dreg      equ       $800003     *Adresse på ACIA'ens data register

adc            equ       $80000      *Adresse for A/D-konverter

*ds            equ       $100001     *Adresse for digitale modstande

*ASCII værdier
*et            equ       %00110001   *Værdi for tegnet "1"
*to            equ       %00110010   *Værdi for tegnet "2"
*tre           equ       %00110011   *Værdi for tegnet "3"
*fire          equ       %00110100   *Værdi for tegnet "4"
fem            equ       %00110101   *Værdi for tegnet "5"
seks           equ       %00110110   *Værdi for tegnet "6"
syv            equ       %00110111   *Værdi for tegnet "7"
```

```
M               equ       %11001101  *Værdi for tegnet "M" tilpasset museprotokollen

*************************************************************
*                                                           *
*               Globale variable                            *
*                                                           *
*************************************************************
kalint          equ       $60000     *Variabel til beregning af offset, byte

taeldata        equ       $60002     *Tæller til kalibreringsdata, word

*digtael        equ       $60004     *Tæller til kalibrering, long

option          equ       $60008     *Tæller for "behandling af data", byte

muspak1         equ       $60009     *Første færdige musepakke, byte
muspak2         equ       $6000A     *Anden færdige musepakke, byte
muspak3         equ       $6000B     *Tredje færdige musepakke, byte

tael            equ       $60010     *Tæller til delay, long

taelint1        equ       $60014     *Tæller for antal INT1, word
taelint2        equ       $60016     *Tæller for antal INT2, word

taersk1         equ       $60018     *Tærskelværdi for kanal 1, byte
taersk2         equ       $60019     *Tærskelværdi for kanal 2, byte
taersk3         equ       $6001A     *Tærskelværdi for kanal 3, byte
taersk4         equ       $6001B     *Tærskelværdi for kanal 4, byte

sendm           equ       $6001C     *Tæller til antal sendte M'er, word

offset1         equ       $6001E     *Offset for kanal 1, long
offset2         equ       $60022     *Offset for kanal 2, long
offset3         equ       $60026     *Offset for kanal 3, long
offset4         equ       $6002A     *Offset for kanal 4, long

*digmod1        equ       $6002E     *Placering af variabler
*digmod2        equ       $6002F     *for værdier i de
*digmod3        equ       $60030     *digitale modstande. Bytes
*digmod4        equ       $60031

*digdata        equ       $61000     *Startadresse for data til digitale modstande

data1           equ       $50000     *Startadresse for data til kanal 1
data2           equ       $53000     *Startadresse for data til kanal 2
data3           equ       $56000     *Startadresse for data til kanal 3
data4           equ       $59000     *Startadresse for data til kanal 4

*************************************************************
*                                                           *
*               Initialisering                              *
*                                                           *
*************************************************************
                move.w    #$2700,sr              *Deaktiver alle interrupts

                lea       $7FF00,a7              *Initialiser stack-pointer

*ACIA initialiseres
                move.b    #%00000011,acia_kreg   *Reset ACIA.
                move.b    #%10010110,acia_kreg   *Sæt ACIA til
*                                                *modtage interrupt

*Digitale modstande indstilles
*               move.b    #ref,digmod1           *Referenceværdien
*               move.b    #ref,digmod2           *indlæses i de
```

```
*               move.b    #ref,digmod3          *digitale modstandes
*               move.b    #ref,digmod4          *variable
*               bsr       INDL_MODST

*Sæt offset- og tærskelværdier
                move.l    #$7F,offset1          *Sæt offset værdierne
                move.l    #$7F,offset2          *til default midt i
                move.l    #$7F,offset3          *spektret
                move.l    #$7F,offset4

                move.b    #30,taersk1           *Sæt default tærskelværdi
                move.b    #30,taersk2
                move.b    #30,taersk3
                move.b    #30,taersk4

                bsr       NULSTIL               *Nulstil brugte registre


*Interrupt rutine 4 aktiveres
                move.w    #$2300,sr             *Aktiver IRQ4 og derover
*                                               *så der interruptes ved
*                                               *modtaget data

IGEN            nop
                jmp       IGEN
                jmp       AFSLUT


************************************************************
*                                                          *
*               Interrupt rutine 1                         *
*                                                          *
************************************************************

INT1
                move.w    #$2700,sr             *Deaktiver alle interrupts

                move.l    #0,d4                 *Nulstil brugte arbejdsregistre
                move.l    #0,d5
                move.l    #0,d6

*Hent data
                move.w    adc,d4                *Hent data fra A/D-konverter

                move.b    d4,d6                 *Hent data fra kanal 2
                rol.w     #8,d4                 *Byt bytes, så kanal 1 kan
*                                               *hentes frem
                move.b    d4,d5                 *Hent data fra kanal 1

*Er variablen "kalint" sat høj
                cmp.b     #0,kalint             *Undersøg bit
                bne       OFF1                  *Hvis høj gå til OFF1

*Træk offset fra data
                sub.l     offset1,d5            *Træk offset fra
                sub.l     offset2,d6

*Ensret data
                btst      #31,d5                *Test for negativt tal
                beq       POSD0                 *Hvis positivt, spring over
                neg.l     d5                    *Ellers, skift fortegn

POSD0           btst      #31,d6                *Test for negativt tal
                beq       POSD1                 *Hvis positivt, spring over
                neg.l     d6                    *Ellers, skift fortegn
```

```
               jmp       POSD1

*kopier data til RAM
OFF1           move.l    d6,(a0)+              *Data gemmes til senere brug
               move.l    d5,(a1)+

*Adder data
POSD1
               add.l     d5,d0                 *Læg til de respektive
               add.l     d6,d1                 *dataregistre

*Læg 1 til tæller
               add.w     #1,taelint1           *Tæl 1 op

*Afslut
               rts


*************************************************************
*                                                           *
*              Interrupt rutine 2                           *
*                                                           *
*************************************************************

INT2
               move.w    #$2700,sr             *Deaktiver alle interrupts

               move.l    #0,d4                 *Nulstil brugte registre
               move.l    #0,d5
               move.l    #0,d6

*Hent data
               move.w    adc,d4                *Hent data fra A/D-konverter

               move.b    d4,d6                 *Hent data fra kanal 4
               rol.w     #8,d4                 *Byt bytes, så kanal 3 kan
*                                              *hentes frem
               move.b    d4,d5                 *Hent data fra kanal 3

*Er variablen "kalint" sat høj
               cmp.b     #0,kalint             *Undersøg bit
               bne       OFF2                  *Hvis høj gå til OFF2

*Træk offset fra data
               sub.l     offset3,d5            *Træk offset fra
               sub.l     offset4,d6

*Ensret data
               btst      #31,d5                *Test for negativt tal
               beq       POSD2                 *Hvis positivt, spring over
               neg.l     d5                    *Ellers, skift fortegn

POSD2          btst      #31,d6                *Test for negativt tal
               beq       POSD3                 *Hvis positivt, spring over
               neg.l     d6                    *Ellers, skift fortegn
               jmp       POSD3

*kopier data til RAM
OFF2           move.l    d6,(a2)+              *Data gemmes til senere brug
               move.l    d5,(a3)+

*Adder data
POSD3
               add.l     d5,d2                 *Læg til de respektive
               add.l     d6,d3                 *dataregistre
```

```
*Læg 1 til tæller
               add.w     #1,taelint2            *Tæl 1 op
*Afslut
               rts



*************************************************************
*                                                           *
*              Interrupt rutine 4                           *
*                                                           *
*************************************************************

INT4           move.w    #$2700,sr              *Deaktiver alle interrupts

*Har ACIA modtaget data?
               move.b    acia_sreg,d5           *Hent ACIA'ens statusregister
               btst      #0,d5                  *Test efter modtaget data
               bne       MODTAGET               *Er test positiv, gå
*                                                *til MODTAGET
               btst      #1,d5                  *Test efter afsendt data
               bne       SENDT                  *Er test positiv, gå til SENDT
               beq       SLUTINT4               *Er test negativ afsluttes

*Styring af musepakker
SENDT          bsr       STYRING                *Kald rutinen STYRING
               jmp       SLUTINT4

MODTAGET       move.b    acia_dreg,d7           *Hent modtaget data
*
*Kalibrering af valgt kanal
*              cmp.b     #et,d7                 *Sammenlign data
*              beq       KANALKAL               *Hvis ens gå til valgt
*                                                *subrutine
*              cmp       #to,d7
*              beq       KANALKAL
*
*              cmp.b     #tre,d7
*              beq       KANALKAL
*
*              cmp       #fire,d7
*              beq       KANALKAL
*
*Kalibrering af offset & tærskelværdi
               cmp       #fem,d7                *Hvis der er modtaget "5",
               beq       OFFKAL                 *startes kalibrering af offset

*Museopstart
               cmp.b     #seks,d7               *Hvis der er modtaget "6",
               beq       STARTMUS               *startes museprogram

*Afslut
               jmp       SLUTINT4               *Afslut hvis intet stemmer

KANALKAL       bsr       KAL_KANAL

               jmp       SLUTINT4

OFFKAL         bsr       KAL_OFF

               jmp       SLUTINT4

STARTMUS       bsr       MUSEOPSTART            *Kald MUSEOPSTART
```

```
SLUTINT4        rts

*************************************************************
*                                                           *
*               Museopstart                                 *
*                                                           *
*************************************************************

MUSEOPSTART
*ACIA re-initialiseres
                move.b    #%00000011,acia_kreg    *Reset ACIA.
                move.b    #%10110110,acia_kreg    *Sæt ACIA til at interrupte
*                                                 *ved modtaget & afsendt data

*Indlæs 0-rapport og nulstil variablen "option"
                move.b    #%11000000,muspak1      *Indlæs 0-pakke
                move.b    #%10000000,muspak2      *Indlæs 0-pakke
                move.b    #%10000000,muspak3      *Indlæs 0-pakke
                bsr       NULSTIL                 *Nulstil anvendte registre

                bsr       DELAY                   *Indsæt pause

*Afsend bekræftelse
                move.b    #syv,acia_dreg          *Send "7" som bekræftelse
                bsr       DELAY                   *Indsæt pause

*Afsend "M" ved CTS
TESTCTS                   move.b   acia_sreg,d5   *Hent ACIA'ens statusregister

                btst      #3,d5                   *Test efter CTS
                beq       STARTSEND               *Er test positiv, gå til
*                                                 *STARTSEND
                bne       TESTCTS                 *Gentag test

STARTSEND       move.w    #0,sendm                *Nulstil tæller "sendm"
VENTSEND        move.b    acia_sreg,d5            *Hent ACIA'ens statusregister
                btst      #1,d5                   *Test om data er afsendt
                beq       VENTSEND                *Hvis ikke, gå til VENTSEND
                move.b    #M,acia_dreg            *Send "M" til ACIA
                add.w     #1,sendm                *Tæl sendm 1 op
                cmp.w     #1200,sendm             *Er der sendt 1200 M'er?
                bls       VENTSEND                *Hvis ikke, send igen

*Enable interrupt rutine 1 og 2 globalt
                move.b    #$20,+8(a7)             *Sæt interruptmaske efter rts,
*                                                 *således at alle IRQ er aktiveret

                rts


*************************************************************
*                                                           *
*               Styring af musepakker                       *
*                                                           *
*************************************************************

STYRING
*Gå til "option" værdi
                cmp.b     #0,option               *Er option 0?
                beq       SEND_MP1                *Send musepakke 1

                cmp.b     #1,option               *Er option 1?
                beq       SEND_MP2                *Send musepakke 2

                cmp.b     #2,option               *Er option 2?
```

```
               beq       SEND_MP3               *Send musepakke 3

               jmp       SLUTSTYRING            *Ellers afslut

SEND_MP1
*Send musepakke1
               move.b    muspak1,acia_dreg      *Send musepakke1 til acia

*Læg 1 til "option" og afslut
               add.b     #1,option              *læg 1 til "option"
               jmp       SLUTSTYRING            *Afslut

SEND_MP2
*Send musepakke2
               move.b    muspak2,acia_dreg      *Send musepakke2 til acia

*Læg 1 til "option" og afslut
               add.b     #1,option              *læg 1 til "option"
               jmp       SLUTSTYRING            *Afslut

SEND_MP3
*Send musepakke3
               move.b    muspak3,acia_dreg      *Send musepakke3 til acia

*Nulstil og start "behandling af data"
               bsr       BEHANDLING             *Kald behandling af data
               bsr       NULSTIL                *Nulstil registre

               jmp       SLUTSTYRING            *Afslut

SLUTSTYRING    rts



************************************************************
*                                                          *
*              Behandling af data                          *
*                                                          *
************************************************************

*Divider dataregistre med "taelint1" eller "taelint2"
BEHANDLING     divu      taelint1,d0            *Divider opsamlede data
               divu      taelint1,d1            *med de respektive
               divu      taelint2,d2            *tællere
               divu      taelint2,d3

               and.l     #$000000FF,d0          *Fjern overskydende data
               and.l     #$000000FF,d1
               and.l     #$000000FF,d2
               and.l     #$000000FF,d3

               move.b    #%11000000,muspak1     *Indlæs 0-pakke som
               move.b    #%10000000,muspak2     *skabelon til brug
               move.b    #%10000000,muspak3     *ved udregninger

*Undersøg for aktivitet
               cmp.b     taersk1,d0             *Er kanal 1 over tærskelværdi?
               bhi       AKTIVD0

               move.b    #0,d0                  *Ellers nulstil kanal 1

AKTIVD0
               cmp.b     taersk2,d1             *Er kanal 2 over tærskelværdi?
               bhi       AKTIVD1
```

```
               move.b    #0,d1                  *Ellers nulstil kanal 2

AKTIVD1
               cmp.b     taersk3,d2             *Er kanal 3 over tærskelværdi?
               bhi       AKTIVD2

               move.b    #0,d2                  *Ellers nulstil kanal 3

AKTIVD2
               cmp.b     taersk4,d3             *Er kanal 4 over tærskelværdi?
               bhi       AKTIVD3

               move.b    #0,d3                  *Ellers nulstil kanal 4

AKTIVD3

*Undersøg for venstreklik
VENSTREKLIK    cmp.b     #0,d0                  *Er der aktivitet på kanal 1?
               beq       HOJREKLIK              *Hvis ikke, undersøg for højreklik
               cmp.b     #0,d1                  *Er der aktivitet på kanal 2?
               beq       HOJREKLIK              *Hvis ikke, undersøg for højreklik

*Pak muserapport
               or.b      #%00100000,muspak1     *Pak musepakke med venstreklik

               rts                              *Afslut rutine

*Undersøg for højreklik
HOJREKLIK      cmp.b     #0,d2                  *Er der aktivitet på kanal 3?
               beq       BEV                    *Hvis ikke, udregn bevægelse
               cmp.b     #0,d3                  *Er der aktivitet på kanal 4?
               beq       BEV                    *Hvis ikke, udregn bevægelse

*Pak muserapport
               or.b      #%00010000,muspak1     *Pak musepakke med højreklik

               rts                              *Afslut rutine

*Omregn bevægelse/pak muserapport
BEV            sub.b     taersk1,d0             *Træk tærskelværdier fra
               sub.b     taersk2,d1             *de respektive dataregistre
               sub.b     taersk3,d2
               sub.b     taersk4,d3

               sub.b     d1,d0                  *Træk kanalerne fra hinanden, så
               sub.b     d3,d2                  *op/ned og venstre/højre bliver til
*                                               *+-x/y i 2's complement

               move.b    d0,muspak2             *Læg x og y værdier i musepakke 2 og 3
               move.b    d2,muspak3

               and.b     #%00111111,muspak2     *Fjern bit 6 og 7
               and.b     #%00111111,muspak3

               or.b      #%10000000,muspak2     *Sæt bit 7 - indikerer
               or.b      #%10000000,muspak3     *8 databit i overførsel

               and.b     #%11000000,d0          *Fjern nederste bit i dataregistre
               and.b     #%11000000,d2

               rol.b     #2,d0                  *Ryk x-værdier på plads
               or.b      d0,muspak1             *Pak i musepakke 1

               rol.b     #4,d2                  *Ryk y-værdier på plads
               or.b      d2,muspak1             *Pak i musepakke 1
```

```
*Afslut
               rts                              *Afslut beregning




*************************************************************
*                                                           *
*              Kalibrering af offset og tærskelværdi        *
*                                                           *
*************************************************************

KAL_OFF
*ACIA re-initialiseres
               move.b    #%00000011,acia_kreg   *Reset ACIA.
               move.b    #%10010110,acia_kreg   *Sæt ACIA til at interrupte
*                                               *ved modtaget data
*Afsend bekræftelse til PC
               move.b    #syv,acia_dreg         *Afsender tegnet bekræftelse til PC

*Indlæs binær max-reference værdi i RAM
*              move.b    refmax,digmod1         *Digitale modstande sættes
*              move.b    refmax,digmod2         *til max. forstærkning
*              move.b    refmax,digmod3
*              move.b    refmax,digmod4

*Indlæs binære max-værdier i modstande
*              bsr       INDL_MODST             *Kald subrutine INDL_MODST

*Nulstil registre, "taelint" og "taelint2"
               bsr       NULSTIL                *Kald subrutine NULSTIL

*Variablen "kalint" sættes høj
               move.b    #1,kalint              *Værdien 1 lægges i "kalint"
               lea       data1,a0               *Der allokeres plads til data
               lea       data2,a1               *til kalibrering
               lea       data3,a2
               lea       data4,a3

*Interrupt 1 og 2 enables
               move.w    #$2000,sr              *Aktiver alle interrupts

*Kør 3000 interrupts
TAELLER1       cmp.w     #3000,taelint1
               beq       DISABLE1
               jmp       TAELLER1

*Interrupt 1 og 2 disables
DISABLE1       move.w    #$2700,sr              *Deaktiver interrupts

*Divider data med "taelint1" og "taelint2"
               divu      taelint1,d0            *Offset værdier findes
               divu      taelint1,d1
               divu      taelint2,d2
               divu      taelint2,d3

               and.l     #$000000FF,d0          *Fjern overflødige data
               and.l     #$000000FF,d1
               and.l     #$000000FF,d2
               and.l     #$000000FF,d3

*Gem offset værdier i offset variable
               move.l    d0,offset1             *Offset værdierne gemmes
```

```
                move.l    d1,offset2            *i variable
                move.l    d2,offset3
                move.l    d3,offset4

*Træk offset værdi fra data gemt i RAM
                lea       data1,a0              *Find adresser til data
                lea       data2,a1
                lea       data3,a2
                lea       data4,a3


                move.w    #0,taeldata
SUBOFFSET       move.l    (a0),d0               *Hent data til dataregistre
                move.l    (a1),d1
                move.l    (a2),d2
                move.l    (a3),d3

                sub.l     offset1,d0            *Træk offset fra
                sub.l     offset2,d1
                sub.l     offset3,d2
                sub.l     offset4,d3

*Ensret data gemt i RAM
                btst      #7,d0                 *Test fortegn
                beq       POS_D0
                neg.l     d0

POS_D0          btst      #7,d1                 *Test fortegn
                beq       POS_D1
                neg.l     d1

POS_D1          btst      #7,d2                 *Test fortegn
                beq       POS_D2
                neg.l     d2

POS_D2          btst      #7,d3                 *Test fortegn
                beq       POS_D3
                neg.l     d3

POS_D3

                move.l    d0,(a0)+              *Gem data i RAM
                move.l    d1,(a1)+
                move.l    d2,(a2)+
                move.l    d3,(a3)+

                add.w     #1,taeldata           *Tæl 1 op
                cmp.w     #3000,taeldata        *Kør igennem alle 3000 datasæt
                bne       SUBOFFSET

                lea       data1,a0              *Find adresser til data
                lea       data2,a1
                lea       data3,a2
                lea       data4,a3

                move.w    #0,taeldata           *Nulstil datatæller

                move.l    #0,d0                 *Nulstil dataregistre
                move.l    #0,d1
                move.l    #0,d2
                move.l    #0,d3

*Adder data gemt i RAM
ADDKALDATA      add.l     (a0)+,d0              *Hent data fra RAM og
                add.l     (a1)+,d1              *læg til dataregistre
```

```
               add.l     (a2)+,d2
               add.l     (a3)+,d3

               add.w     #1,taeldata

               cmp.w     #3000,taeldata          *Kør igennem alle 3000 data
               bne       ADDKALDATA

*Divider data med "taelint1" og "taelint2"
               divu      taelint1,d0             *Tærskel værdi findes
               divu      taelint1,d1
               divu      taelint2,d2
               divu      taelint2,d3

*Indsæt margin
               add.b     #5,d0                   *Indsæt margin
               add.b     #5,d1
               add.b     #5,d2
               add.b     #5,d3

*Gem data som tærskel variable
               move.b    d0,taersk1              *Tærskel værdierne gemmes
               move.b    d1,taersk2              *i variable
               move.b    d2,taersk3
               move.b    d3,taersk4

*Afsend bekræftelse til PC
VENT2SEND      move.b    acia_sreg,d5            *Hent ACIA'ens statusregister
               btst      #1,d5                   *Test om sende buffer er tom
               beq       VENT2SEND               *Hvis ikke, gå til VENT2SEND

               move.b    #syv,acia_dreg          *Afsender tegnet "7" til PC

*Nulstil + afslut
               bsr       NULSTIL
               rts


*************************************************************
*                                                           *
*              Kalibrering af valgt kanal                   *
*                                                           *
*************************************************************
*
*ACIA re-initialiseres
*KAL_KANAL     move.b    #%00000011,acia_kreg    *Reset ACIA.
*              move.b    #%10010110,acia_kreg    *Sæt ACIA til at interrupte
*                                                *ved modtaget data
*
*Afsend bekræftelse til PC
*              move.b    #syv,acia_dreg          *Afsender tegnet "7" til PC
*
*Aktivering af aktuel kanal
*              cmp.b     #et,d7                  *Sammenlign data
*              beq       INDLAES1                *Hvis ens gå til valgt
*                                                *subrutine
*              cmp.b     #to,d7
*              beq       INDLAES2
*
*              cmp.b     #tre,d7
*              beq       INDLAES3
*
*              cmp.b     #fire,d7
*              beq       INDLAES4
*
```

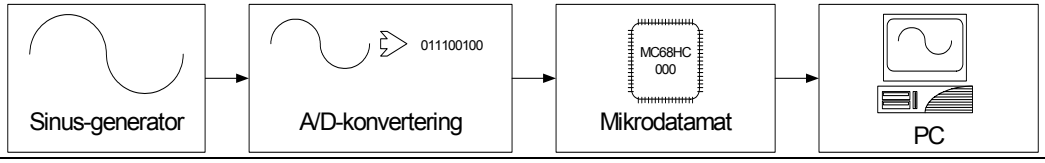
```
*               bsr       NULSTIL                 *Ved fejl nulstil og afslut
*               rts
*
*Indlæs binær reference værdi i RAM/Indlæs binære værdier i modstande
*INDLAES1       move.b    ref,digmod1             *Indlæs reference værdi
*               jmp       INDLAES
*INDLAES2       move.b    ref,digmod2
*               jmp       INDLAES
*INDLAES3       move.b    ref,digmod3
*               jmp       INDLAES
*INDLAES4       move.b    ref,digmod4
*               jmp       INDLAES
*INDLAES        bsr       INDL_MODST              *Indlæs binære værdier i modstande
*
*Nulstil registre "taelint1" og "taelint2"
*               bsr       NULSTIL                 *Nulstilling
*
*Interrupt 1 og 2 enables
*               move.w    #$2000,sr               *Aktiver IRQ1 og derover
*
*Modtag 3000 interrupts
*TAELLER2       cmp       #3000,taelint1
*               beq       DISABLE2
*               jmp       TAELLER2
*
*Interrupt 1 og 2 disables
*DISABLE2       move.w    #$2700,sr               *Deaktiver interrupts
*
*Divider adresser med "taelint1" og "taelint2"
*               divu      taelint1,d0             *Divider med tællerværdi
*               divu      taelint1,d1
*               divu      taelint2,d2
*               divu      taelint2,d3
*
*Udregn ny binær værdi til modstande/Gem ny binær værdi i RAM
*               cmp.b     #et,d7                  *Sammenlign data
*               beq       BINAER1                 *Hvis ens gå til valgt
*                                                 *subrutine
*               cmp       #to,d7
*               beq       BINAER2
*
*               cmp.b     #tre,d7
*               beq       BINAER3
*
*               cmp       #fire,d7
*               beq       BINAER4
*
*BINAER1
*               move.w    #25500,d6               *Værdien 25500 fyttes til d6
*               divu      d0,d6                   *Forholdet mellem aktuelt
*                                                 *niveau og maxniveau
*               move.b    #100,d0                 *Ligningen
*               sub.w     d0,d6                   *x=((d6-100)255)/1000 løses
*               mulu      #255,d6                 *jf. ligning 3.2
*               move.w    #1000,d0
*               divu      d0,d6
*               move.b    d6,digmod1              *Værdien lægges i variablen
*               jmp       STOPBIN
*
*BINAER2
*               move.w    #25500,d6
*               divu      d1,d6
*               move.b    #100,d0
*               sub.w     d1,d6
*               mulu      #255,d6
```

```
*               move.w    #1000,d1
*               divu      d1,d6
*               move.b    d6,digmod2
*               jmp       STOPBIN
*
*BINAER3
*               move.w    #25500,d6
*               divu      d2,d6
*               move.b    #100,d2
*               sub.w     d2,d6
*               mulu      #255,d6
*               move.w    #1000,d2
*               divu      d2,d6
*               move.b    d6,digmod3
*               jmp       STOPBIN
*
*BINAER4
*               move.w    #25500,d6
*               divu      d3,d6
*               move.b    #100,d3
*               sub.w     d3,d6
*               mulu      #255,d6
*               move.w    #1000,d3
*               divu      d3,d6
*               move.b    d6,digmod4
*               jmp       STOPBIN
*
*STOPBIN
*
*Indlæs binær værdier i modstande
*               bsr       INDL_MODST
*
*Afsend bekræftelse til PC
*               move.b    #syv,acia_dreg          *Afsender tegnet "7" til PC
*
*Nulstil + afslut
*               bsr       NULSTIL
*               rts
*
*
*INDL_MODST
*               move.b    #0,digtael
*               move.l    #0,d5
*
*               lea       digdata,a0              *Opsæt adressestart
*
*               move.b    #%00001000,(a0)+        *Lav pakke til stackbit
*               move.b    #%00001100,(a0)+
*
*               move.b    digmod2,d0              *Flyt binærværdier til
*                                                 *arbejds-dataregistre
*               move.b    digmod4,d1
*
*PAKDIG
*               move.l    #0,d2
*               move.l    #0,d3
*               move.l    #0,d4
*               btst      #0,digtael              *Test om tæller er lige/ulige
*               bne       PAKCLK1                 *Hvis ulige gå til PAKCLK1
*                                                 *ellers PAKCLK0
*
*PAKCLK0        move.b    #%00001000,d2           *Klokken og RST lægges i d2
*               jmp       TEST16
*
*PAKCLK                   move.b   #%00001100,d2  *Klokken lægges i d2
```

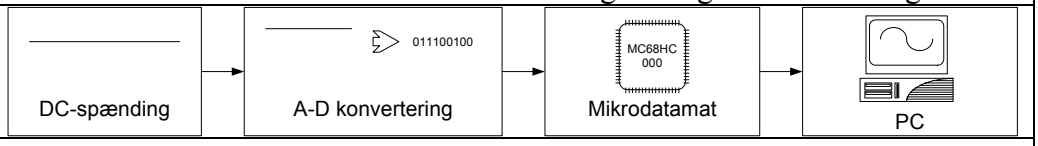
```
*
*TEST16                 cmp.b   #16,digtael    *Undersøg om tælleren er 16
*               beq     FLYT1OG3               *16 gå til FLYT1OG3
*               jmp     PAK0                   *Ikke 16 gå til PAK0
*
*FLYT1OG3       move.b  digmod1,d0             *Flyt binærværdier til
*                                              *arbejds-dataregistre
*               move.b  digmod3,d1
*               move.l  #0,d5                  *Nulstil variablen bit
*
*PAK0           btst    d5,d0                  *Undersøg bit X
*               beq     PAKNUL0                *Hvis 0 gå til PAKNUL0
*                                              *ellers PAKET0
*PAKET0                 move.b  #1,d3          *Bit X lægges i d3
*               jmp     PAK1
*
*PAKNUL0                move.b  #0,d3          *Bit X lægges i d3
*
*PAK1           btst    d5,d1                  *Undersøg bit X
*               beq     PAKNUL1                *Hvis 0 gå til PAKNUL1
*                                              *ellers PAKET1
*
*PAKET1                 move.b  #2,d4          *Bit X lægges i d4
*               jmp     OR
*
*PAKNUL1                move.b  #0,d4          *Bit X lægges i d4
*
*OR             or      d2,d3                  *d2 og d3 OR'es
*               or      d3,d4                  *d3 og d4 OR'es
*               move.b  d4,(a0)+               *Resultatet lagres i RAM
*
*               btst    #0,digtael             *Test om tæller er lige/ulige
*               beq     ADDDIG                 *Hvis lige gå til DIGTAEL1
*                                              *ellers fortsæt
*               add.b   #1,d5                  *Læg 1 til variablen d5
*
*ADDDIG                 add.b   #1,digtael     *Læg 1 til tælleren
*
*               cmp.b   #32,digtael            *Undersøg om tæller er 32
*               bne     PAKDIG                 *Hvis ikke gå til PAKDIG
*                                              *Hvis 32 fortsæt
*
*               lea     digdata,a0
*               move.b  #0,digtael
*
*AFSENDDIG
*               move.b  (a0)+,ds
*               add.b   #1,digtael
*               cmp.b   #34,digtael
*               bne     AFSENDDIG
*
*               move.b  #0,ds
*
*               rts


*Nulstil dataregistre
NULSTIL
                move.l  #0,d0                  *Nulstil dataregistre
                move.l  #0,d1
                move.l  #0,d2
                move.l  #0,d3
                move.b  #0,option              *Nulstil "option"
                move.w  #0,taelint1            *Nulstil interrupttællere
                move.w  #0,taelint2
```

```
                move.w    #0,sendm              *Nulstil "sendm" tæller
                move.b    #0,kalint             *Nulstil kalint

                rts                             *Afslut


*Delayrutine
DELAY           move.l    #0,tael               *Nulstil tæller
TAELDELAY       add.l     #1,tael               *Tæl 1 op
                cmp.l     #150000,tael          *Tæl til 150000
                bne       TAELDELAY
                rts                             *Afslut

AFSLUT          end       $41000                *Afslut program
```

I.2 Test af offsetkalibrering

Formål	At kontrollere hvorvidt offsetkalibreringen fungerer efter hensigten.
Opstilling	
Instrumenter	• Spændingsforsyning: Delta Elektronika E030–3 LBNR: 08500 Tage Juul Elektronic LBNR: 07379 • Funktionsgenerator: Wavetek Model 116 LBNR: 06924 • Oscilloskop: Agilent 54621A LBNR: 33865 • PC
Fremgangsmåde	• Kalibrering af offset- og tærskelværdier vælges i PC programmet. • Med oscilloskop måles offset værdierne til ca. 2,5V på AD-konverterens 4 indgangskanaler. • Hukommelsen på mikrodatamaten undersøges for offsetværdierne .
Resultater	Hukommelsesadresserne 6001Eh -22h -26h og -2Ah undersøges og værdierne blev aflæst til at ligge i intervallet 7Dh og 80h.
Kommentar	De aflæste værdier er fundet tilfredsstillende, idet intervallet svarer til offsetværdier i intervallet mellem 2,45V og 2,51V. Det kan konkluderes at offsetkalibreringen fungerer korrekt.

I.3 Test af tærskelværdi kalibrering

Formål	At kontrollere hvorvidt tærskelværdikalibreringen fungerer efter hensigten.
Opstilling	
Instrumenter	• Spændingsforsyning: Delta Elektronika E030–3 LBNR: 08500 Tage Juul Elektronic LBNR: 07379 • Funktionsgenerator: Wavetek Model 116 LBNR: 06924 • Oscilloskop: Agilent 54621A LBNR: 33865 • PC
Fremgangsmåde	• Kalibrering af offset- og tærskelværdier vælges i PC programmet.

	• Der sættes 5V ind på AD-konverterens 4 indgangskanaler. • Hukommelsen på mikrodatamaten undersøges for tærskelværdierne. • Efterfølgende sættes 2,5V ind på AD-konverterens 4 indgangskanaler. • Hukommelsen på mikrodatamaten undersøges for tærskelværdierne.
Resultater	Hukommelsesadresserne 60018h -19h -1Ah og -1Bh undersøges og værdierne FFh og 7Fh blev aflæst, for henholdsvis 5V og 2,5V indgangssignal.
Kommentar	Tærskelværdi kalibreringen må konkluderes at fungere korrekt, idet de målte tærskelværdier stemmer overens med ingangssignalerne.

I.4 Testkode til indlæsning af sinussignal

```
*******************************************************
*                                                     *
*           Interrupt tabel                           *
*                                                     *
*******************************************************
            org         $400C8                  *Placering af INT1
            jmp         INT1                    *Gå til interrupt rutine 1

            org         $400D0                  *Placering af INT2
            jmp         INT2                    *Gå til interrupt rutine 2

            org         $400D8                  *Placering af INT3
            jmp         INT2                    *Gå til interrupt rutine 2

            org         $400E0                  *Placering af INT4
            jmp         INT4                    *Gå til interrupt rutine 4

            org         $400E8                  *Placering af INT5
            jmp         INT4                    *Gå til interrupt rutine 4

            org         $400F0                  *Placering af INT6
            jmp         INT4                    *Gå til interrupt rutine 4


*******************************************************
*                                                     *
*           Konstanter                                *
*                                                     *
*******************************************************

            org         $41000                  *Startadresse for program

acia_kreg   equ         $800001                 *Adresse på ACIA'ens kontrol register
acia_dreg   equ         $800003                 *Adresse på ACIA'ens data register

adc         equ         $80000                  *Adresse på A/D-konverter

data        equ         $60000

taelint1    equ         $57000
taelint2    equ         $57004
taelint4    equ         $57008
taelmax     equ         $5700C
```

```
*********************************************************
*                                                       *
*           Initialisering                              *
*                                                       *
*********************************************************
*ACIA initialiseres
            move.b      #%00000011,acia_kreg    *Reset ACIA.
            move.b      #%00110101,acia_kreg    *Sæt ACIA til
*                                               *sende interrupt

*Interrupt rutine 4 aktiveres
            move.w      #$2000,sr               *Aktiver IRQ1 og derover

            lea         data,a0                 *Sæt adresse til datalagring
            lea         data,a1

            move.l      #0,taelint1             *Nulstil interrupttællere
            move.l      #0,taelint2
            move.l      #0,taelint4

            move.b      #0,acia_dreg            *Send 0 til ACIA. Starter sendecyklus

MAAL        move.l      #6000,d0                *Test om der har været 6000 målinger
            cmp.l       taelint1,d0
            bne         MAAL

            move.w      #$2300,sr               *Deaktiver ADC interrupt efter 6000
                                                *målinger

SEND        move.l      #6000,d1                *Test efter 6000 sendte tegn.
            cmp.l       taelint4,d1
            bne         SEND

            move.w      #$2700,sr               *Deaktiver alle interrupts

IGEN        nop                                 *Efter konvertering gåes i uendelig
                                                *løkke
            jmp         igen

            jmp         AFSLUT


*Udkommenter de kanaler, der ikke testes på

INT1        move.w      adc,d4                  *Hent data fra A/D-konverter
*           move.b      d4,(a0)+                *Læs fra kanal 2
            rol.w       #8,d4                   *Byt øvre og nedre byte
*           move.b      d4,(a0)+                *Læs fra kanal 1
            add.l       #1,taelint1
*Afslut
            rte

INT2        move.w      adc,d4                  *Hent data fra A/D-konverter
            move.b      d4,(a0)+                *Læs fra kanal 4
            rol.w       #8,d4                   *Byt øvre og nedre byte
*           move.b      d4,(a0)+                *Læs fra kanal 3
            add.l       #1,taelint2
*Afslut
            rte

INT4        move.b      (a1)+,acia_dreg         *Send næste data til acia
            add.l       #1,taelint4
            rte

AFSLUT      end         $41000
```

I.5 Testkode til generering af musepakker

```
*********************************************************
*                                                       *
*           Interrupt tabel                             *
*                                                       *
*********************************************************

            org         $400C8      *Placering af INT1
            bsr         INT1        *Gå til interrupt rutine 1
            rte

            org         $400D0      *Placering af INT2
            bsr         INT2        *Gå til interrupt rutine 2
            rte

            org         $400D8      *Placering af INT3
            bsr         INT2        *Gå til interrupt rutine 2
            rte

            org         $400E0      *Placering af INT4
            bsr         INT4        *Gå til interrupt rutine 4
            rte

            org         $400E8      *Placering af INT5
            bsr         INT4        *Gå til interrupt rutine 4
            rte

            org         $400F0      *Placering af INT6
            bsr         INT4        *Gå til interrupt rutine 4
            rte

*********************************************************
*                                                       *
*           Konstanter                                  *
*                                                       *
*********************************************************

            org         $41000      *Startadresse for program

acia_kreg   equ         $800001     *Adresse på ACIA'ens kontrol register
acia_sreg   equ         $800001     *Adresse på ACIA'ens status register
acia_dreg   equ         $800003     *Adresse på ACIA'ens data register

*ASCII værdier
et          equ         %00110001   *Værdi for tegnet "1"
to          equ         %00110010   *Værdi for tegnet "2"
tre         equ         %00110011   *Værdi for tegnet "3"
fire        equ         %00110100   *Værdi for tegnet "4"
fem         equ         %00110101   *Værdi for tegnet "5"
seks        equ         %00110110   *Værdi for tegnet "6"
syv         equ         %00110111   *Værdi for tegnet "7"
M           equ         %11001101   *Værdi for tegnet "M" tilpasset museprotokollen

*********************************************************
*                                                       *
*           Globale variable                            *
*                                                       *
*********************************************************

option      equ         $60008                  *Tæller for "behandl data"

muspak1     equ         $60009                  *Første færdige musepakke
```

```
muspak2     equ         $6000A                  *Anden færdige musepakke
muspak3     equ         $6000B                  *Tredje færdige musepakke

tael        equ         $60010                  *Tæller til delay rutine

sendm       equ         $60020                  *Tæller til antal sendte M'er

pakvaelg    equ         $60030                  *Indikator for valg af pakke

paktael     equ         $60040                  *Tæller for sendte pakker

*******************************************************
*                                                     *
*           Initialisering                            *
*                                                     *
*******************************************************
            move.w      #$2700,sr               *Deaktiver interrupts

*ACIA initialiseres
            move.b      #%00000011,acia_kreg    *Reset ACIA.
            move.b      #%10010110,acia_kreg    *Sæt ACIA til modtage interrupt


            move.w      #0,pakvaelg             *Nulstil variable
            move.w      #0,paktael
            move.w      #0,sendm

*Interrupt rutine 4 aktiveres
            move.w      #$2300,sr               *Aktiver IRQ4 og derover


PAKCYK      cmp.w       #0,pakvaelg             *Vælg pakke efter pakvaelg
            beq         PAK_1

            cmp.w       #1,pakvaelg
            beq         PAK_2

            cmp.w       #2,pakvaelg
            beq         PAK_3

            cmp.w       #3,pakvaelg
            beq         PAK_4

            jmp         PAKCYK


*Alle pakker flytter 1 pixel:
PAK_1       move.b      #%11000000,muspak1      *Indlæs højre pakke
            move.b      #%10000001,muspak2
            move.b      #%10000000,muspak3
            jmp         PAKCYK

PAK_2       move.b      #%11000000,muspak1      *Indlæs op pakke
            move.b      #%10000000,muspak2
            move.b      #%10000001,muspak3
            jmp         PAKCYK

PAK_3       move.b      #%11000011,muspak1      *Indlæs venstre pakke
            move.b      #%10111111,muspak2
            move.b      #%10000000,muspak3
            jmp         PAKCYK

PAK_4       move.b      #%11001100,muspak1      *Indlæs ned pakke
            move.b      #%10000000,muspak2
            move.b      #%10111111,muspak3
```

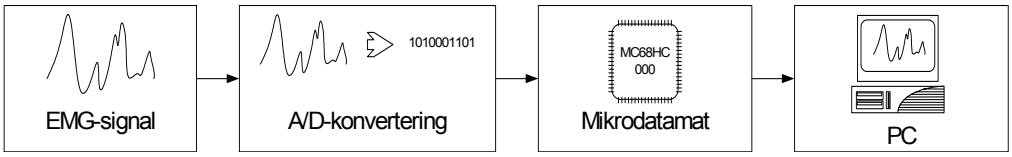
```
            jmp         PAKCYK

IGEN        nop                                 *Holder programmet kørende ved
            jmp         IGEN                    *afslutning
            jmp         AFSLUT


*******************************************************
*                                                     *
*           Interrupt rutiner                         *
*                                                     *
*******************************************************

INT1        rts                                 *Interrupt 1 og 2 skal ikke bruges

INT2        rts



INT4        move.w      #$2700,sr               *Deaktiver alle interrupts

*Har ACIA modtaget data?
            move.b      acia_sreg,d5            *Hent ACIA'ens statusregister
            btst        #0,d5                   *Test efter modtaget data
            bne         MODTAGET                *Er test positiv, gå til MODTAGET
            btst        #1,d5                   *Test efter afsendt data
            beq         SLUTINT4                *Er test negativ afsluttes
            bne         SENDT                   *Er test positiv, gå til SENDT

SENDT
            bsr         STYRING                 *Kald rutinen STYRING
            jmp         SLUTINT4

MODTAGET    move.b      acia_dreg,d7            *Hent modtaget data
            cmp.b       #seks,d7
            beq         MUSEOPSTART

            jmp         SLUTINT4                *Afslut hvis intet stemmer

SLUTINT4    rts

*******************************************************
*                                                     *
*           Museopstart                               *
*                                                     *
*******************************************************

MUSEOPSTART
*ACIA initialiseres
            move.b      #%00000011,acia_kreg    *Reset ACIA.
            move.b      #%10110110,acia_kreg    *Sæt ACIA til at interrupte
*                                               *ved modtaget og afsendt data

*Indlæs 0-rapport og nulstil variablen "taeloption"
            move.b      #%11000000,muspak1      *Indlæs start pakke
            move.b      #%10000001,muspak2
            move.b      #%10000001,muspak3
            move.b      #0,option

            move.b      #syv,acia_dreg          *Send bekræftelse på opstart
            bsr         DELAY

*Afsend "M" ved CTS
TESTCTS     move.b      acia_sreg,d5            *Hent ACIA'ens statusregister
```

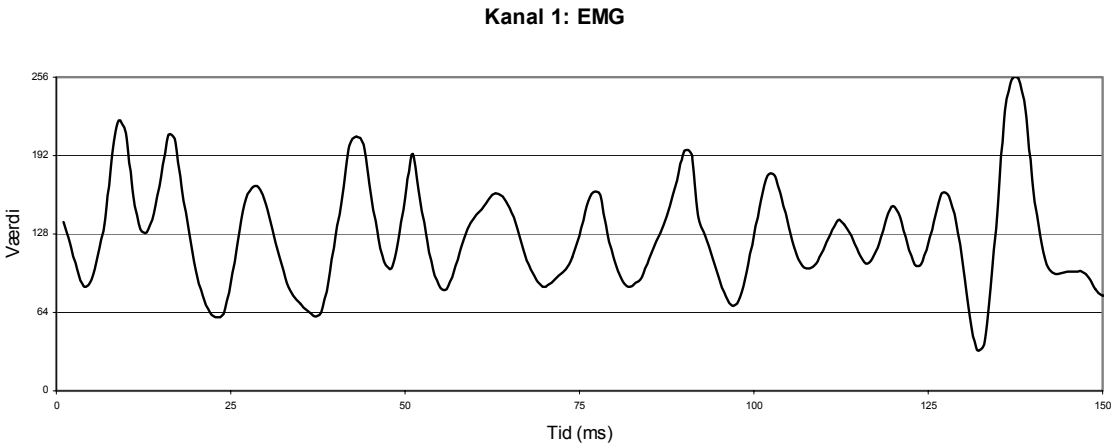
```
            btst        #3,d5                   *Test efter CTS
            beq         STARTSEND               *Er test positiv, gå til STARTSEND
            bne         TESTCTS                 *Gentag test

STARTSEND   move.w      #0,sendm                *Nulstil tæller til sendte M'er
VENTSEND    move.b      acia_sreg,d5            *Hent ACIA'ens statusregister
            btst        #1,d5                   *Test om data er afsendt
            beq         VENTSEND                *Er test negativ, gentag test
AFSEND      move.b      #M,acia_dreg            *Læg "M" i ACIA
            add.w       #1,sendm                *Læg 1 til "M tæller"
            cmp.w       #1200,sendm             *Er der sendt 1200 M'er?
            bls         VENTSEND                *Hvis ikke, send igen
            rts                                 *Afslut interrupt rutine


*********************************************************
*                                                       *
*           Styring af musepakker                       *
*                                                       *
*********************************************************

STYRING
*Send næste musepakke til ACIA
            cmp.b       #0,option               *Sammenlign "option"
*                                               *med værdien 0
            beq         SEND_MP1                *Send musepakke1
            cmp.b       #1,option
            beq         SEND_MP2
            cmp.b       #2,option
            beq         SEND_MP3
            jmp         SLUTSTYRING

SEND_MP1
*Læg 1 til "option" og afslut
            move.b      muspak1,acia_dreg       *Send musepakke1 til acia
            add.b       #1,option               *læg 1 til "option"
            jmp         SLUTSTYRING

SEND_MP2
*Læg 1 til "taeloption" og afslut
            move.b      muspak2,acia_dreg       *Send musepakke2 til acia
            add.b       #1,option               *læg 1 til "option"
            jmp         SLUTSTYRING

SEND_MP3
*Nulstil "taeloption" og start behandling af data
            move.b      muspak3,acia_dreg       *Send musepakke3 til acia
            move.b      #0,option               *nulstil "option"

            add.w       #1,paktael              *Læg 1 til pakketælleren

            cmp.w       #100,paktael            *Er der sendt 100 pakker?
            bls         SLUTSTYRING             *Hvis ikke, afslut og send en mere

            move.w      #0,paktael              *Nulstil pakketæller
            add.w       #1,pakvaelg             *Læg 1 til pakkevælger

            cmp.w       #4,pakvaelg             *Er musen nået rundt i en firkant?
            bne         SLUTSTYRING             *Hvis ikke, afslut

            move.w      #0,pakvaelg             *Ellers nulstil tællere
            move.w      #0,paktael

            jmp         SLUTSTYRING
```

```
SLUTSTYRING rts                             *Afslut

DELAY       move.l      #0,tael             *Nulstil tæller

TAELDELAY   add.l       #1,tael             *Læg 1 til tæller
            cmp.l       #150000,tael        *Er den kommer til 150000?
            bne         TAELDELAY           *Hvis ikke, tæl igen
            rts

AFSLUT      end         $41000
```

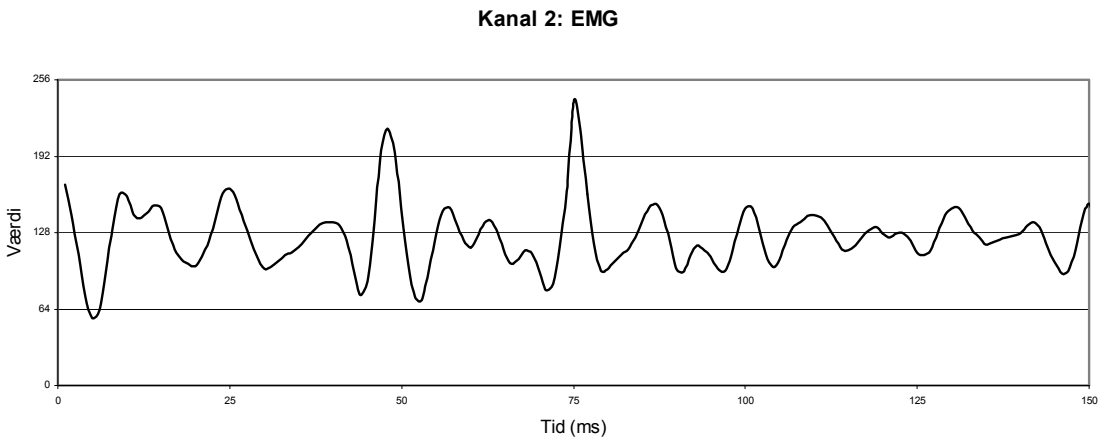
J Tests af samlet system

J.1 Test af hardware

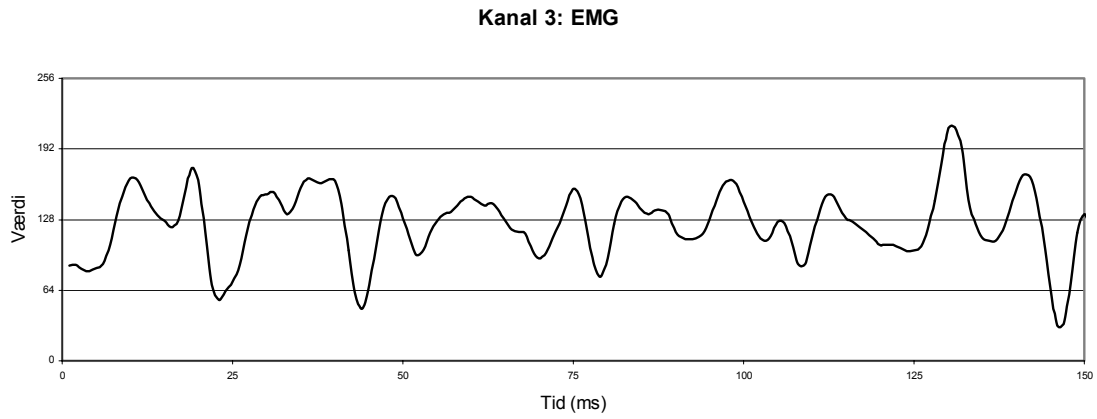
Formål	At kontrollere at et EMG signal kan læses ind i mikrodatamaten og derefter udlæses, dette forsøg udføres for at sikre at EMG signalet kan læses ind i mikrodatamaten. Denne skal ses som en test af den samlede hardware.
Opstilling	
Instrumenter	• Spændingsforsyning: Delta Elektronika E030–3 LBNR: 08500 Tage Juul Elektronic LBNR: 07379 • Funktionsgenerator: Wavetek Model 116 LBNR: 06924 • Oscilloskop: Agilent 54621A LBNR: 33865 • PC
Fremgangsmåde	• Der indlæses programkode i mikrodatamaten, der læser én kanals 6 sekunders sampling ud til PC'en. • EMG signal sættes på den valgte kanal. • Programmet initialiseres og overfører data for kanalen ud til PC'en. • Data behandles og undersøges for lighed med et EMG signal.
Resultater	Resultaterne for de fire kanaler ses i figurerne J.19 til J.22
Kommentar	De samme egenskaber som i Test af A/D-konvertering gælder for denne test. Resultaterne er tilfredsstillende, idet dataene der er ud læst fra mikrodatamaten stemmer over ens med det forventede EMG signal, efter filtrering.



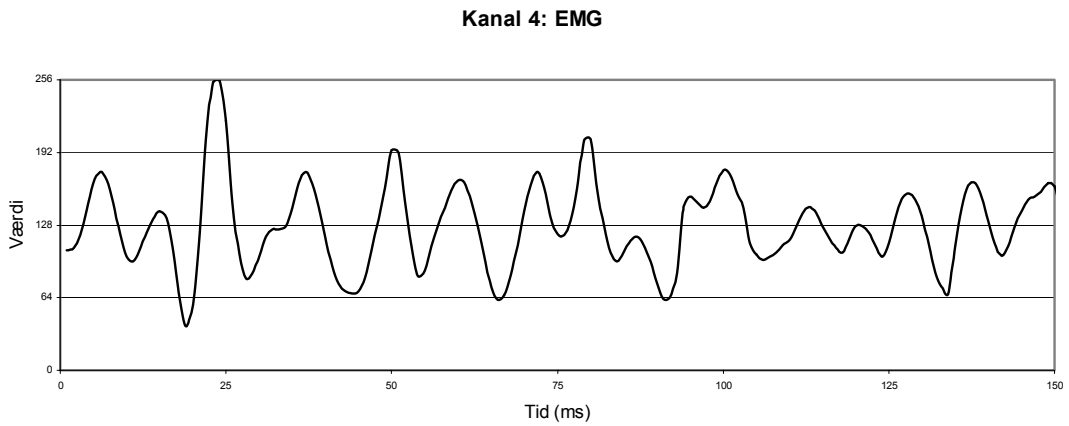
Figur J.19: Test af kanal 1



Figur J.20: Test af kanal 2

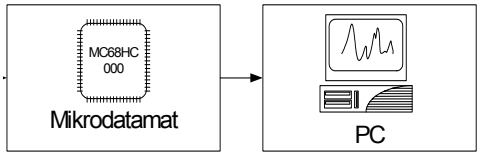


Figur J.21: Test af kanal3

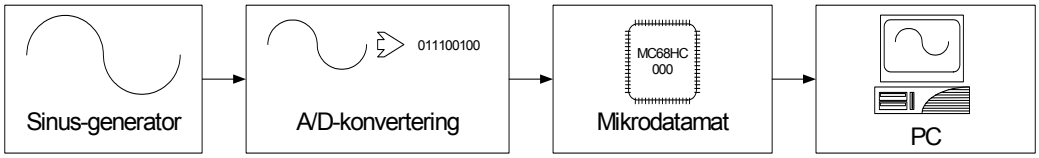


Figur J.22: Test af kanal 4

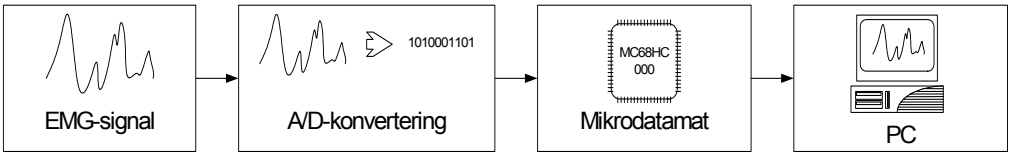
J.2 Test af musepakke software

Formål	Ved hjælp af et testprogram at få musemarkøren til at bevæge sig i en firkant på skærmen.
Opstilling	
Instrumenter	• Spændingsforsyning: Delta Elektronika E030–3 LBNR: 08500 Tage Juul Elektronic LBNR: 07379 • Funktionsgenerator: Wavetek Model 116 LBNR: 06924 • Oscilloskop: Agilent 54621A LBNR: 33865 • PC
Fremgangsmåde	• Lav testprogram der skal få musen til at bevæge sig i en firkant på skærmen (Appendiks I.5). • Download testprogram i mikrodatamaten. • Start testprogram. • Søg efter ny hardware. • Optag skærm.
Resultater	Resultatet må siges at være tilfredsstillende, da musen bevægede sig rundt i en firkant på skærmen.
Kommentar	Som dokumentation er der vedlagt et video klip, som viser testen [CD\Test\musetest.avi / *.mpg]. På videoen ses hvordan musesoftwaren indlæses i mikrodatamaten og startes ved hjælp af PC-programmet. Derefter startes søgningen efter ny hardware, hvor datamaten identificeres som en mus, og får musepilen til at bevæge sig i en firkant. Derved viser testen, at softwaren til generering af musepakker virker. På klippet ses fremgangsmåden for indlæsning af software, samt hardwaresøgning, der resulterer i at Mikrodatamaten identificeres som en mus af Windows®. Der er på billedet indlagt fire kontrolpunkter med 100 pixels imellem sig. Softwaren i mikrodatamaten er indstillet til at flytte pilen i en firkant, med hundrede pixels i hver retning. De fire punkter kan derfor anvendes til at kontrollere om softwaren flytter pilen det rigtige antal pixels. Som det ses i klippet virker softwaren korrekt. Musepilen flytter nøjagtig 100 pixels i hver retning. Der kan derudfra konkluderes at musesoftwaren til generering af musepakker fungerer korrekt.

J.3 Test af kalibrering

Formål	At undersøge om mikrodatamaten vha. de digitale modstande kan regulere forstærkningen i den analoge del af systemet.
Opstilling	
Instrumenter	• Spændingsforsyning: Delta Elektronika E030–3 LBNR: 08500 Tage Juul Elektronic LBNR: 07379 • Funktionsgenerator: Wavetek Model 116 LBNR: 06924 • Oscilloskop: Agilent 54621A LBNR: 33865 • PC
Fremgangsmåde	• På indgangen af instrumenteringsforstærkeren påtrykkes en sinusformet spænding på $1m\hat{V}$ med en frekvens på 100Hz. • Det forsøges at ændre værdien af den digitale modstand vha. mikrodatamaten $(20k\Omega,\ 50k\Omega\ og\ 100k\Omega)$.
Resultater	Ikke anvendelige da der ikke umiddelbart var nogen sammenhænge mellem den indlæste modstandsværdi og den reelle forstærkning.
Kommentar	Startværdien i den digitale modstand er på $50k\Omega$, hvilket betyder, at der på indgangen af den digitale modstand burde være $500 \cdot 6 \cdot 1m\hat{V} = 3\hat{V}$. Dette var også tilfældet, men forsøgtes det at ændre værdien af modstanden ændrede forstærkningen sig ikke. Det kan hermed konkluderes, at digital regulering af forstærkningen ikke er muligt i det pågældende system.

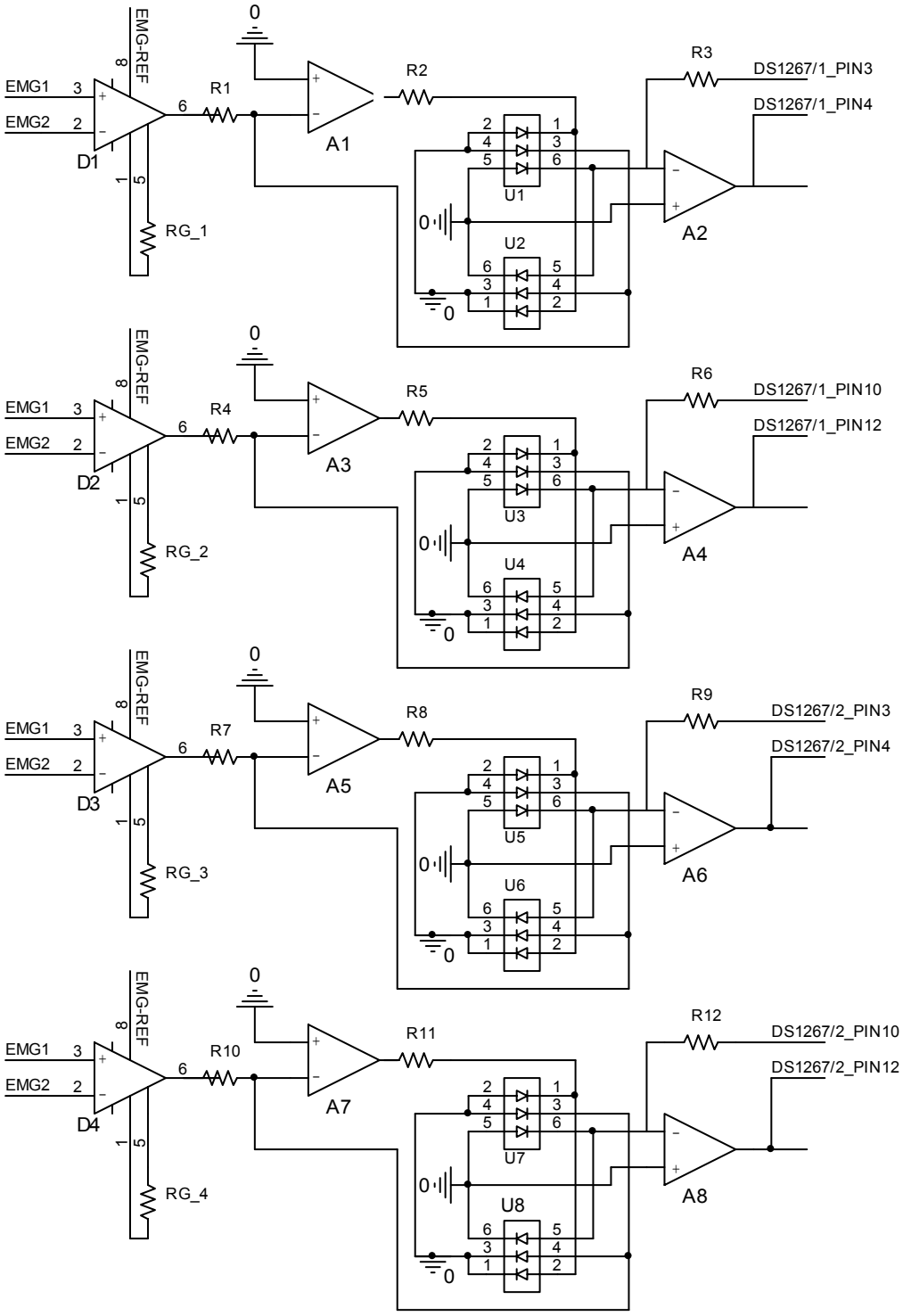
J.4 Test af samlet system

Formål	Ved hjælp af elektroder skal musen kunne bevæges rundt på skærmen. Det skal undersøges om det er muligt at både højre og venstre klikke. Dette gennemføres uden brug af den digitale kalibrering, i dette forsøg gøres der brug af manuel kalibrering.
Opstilling	
Instrumenter	• Spændingsforsyning: Delta Elektronika E030–3 LBNR: 08500 Tage Juul Elektronic LBNR: 07379 • Funktionsgenerator: Wavetek Model 116 LBNR: 06924 • Oscilloskop: Agilent 54621A LBNR: 33865 • PC
Fremgangsmåde	• Program uploades til mikrodatamat, ved hjælp af PC-program. • Bruger påføres elektroder på arme og ben. • Forstærkningen indstilles til en passende værdi. • Program startes.

	• Der søges efter ny hardware. • Herefter testes om musen kan bruges.
Resultater	Det kan konkluderes at det er muligt, at styre musen og det samtidigt er muligt at både højre- og venstreklikke.
Kommentar	Det skal tilføjes, at det kræver en del øvelse at bruge musen, da den kan være rimelig følsom. Dette kan dog justeres i forstærkningen, eller gennem softwarens tærskelværdi, så det passer til den pågældende bruger.

K Diagrammer/komponentliste

K.1 Forstærkning

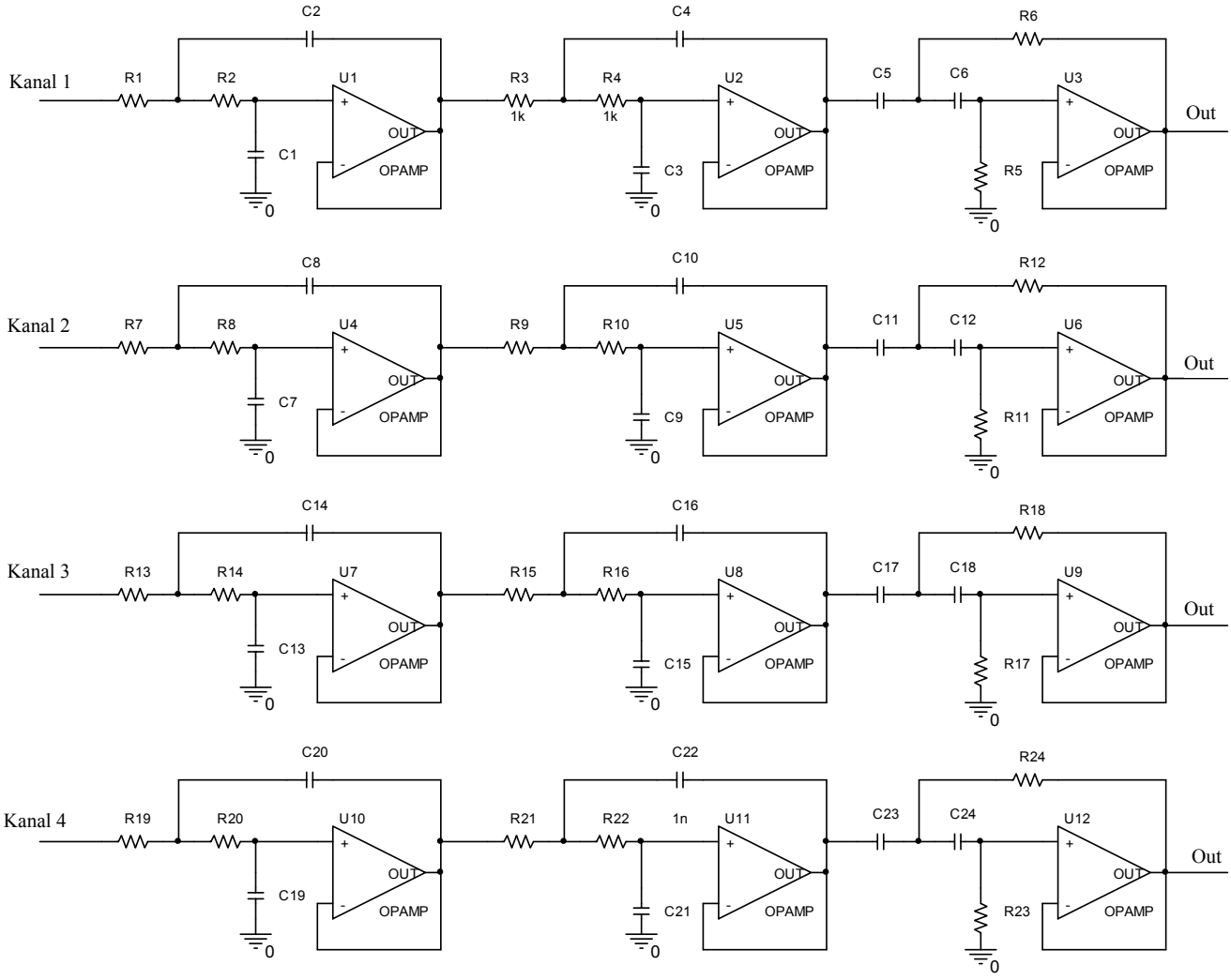


Figur K.1: forstærkning og galvaniskadskillelse

Komponentværdier:

D1 - D4:	AD620	R3, R6	
		R9, R12:	$10\,k\Omega$
RG 1 - RG4:	$100\,\Omega$		
		A1 - A8:	TLE2072
R1, R4			
R7, R10:	$10\,k\Omega$	U1 - U8:	IL300
R2, R5			
R8, R11:	$10\,k\Omega$		

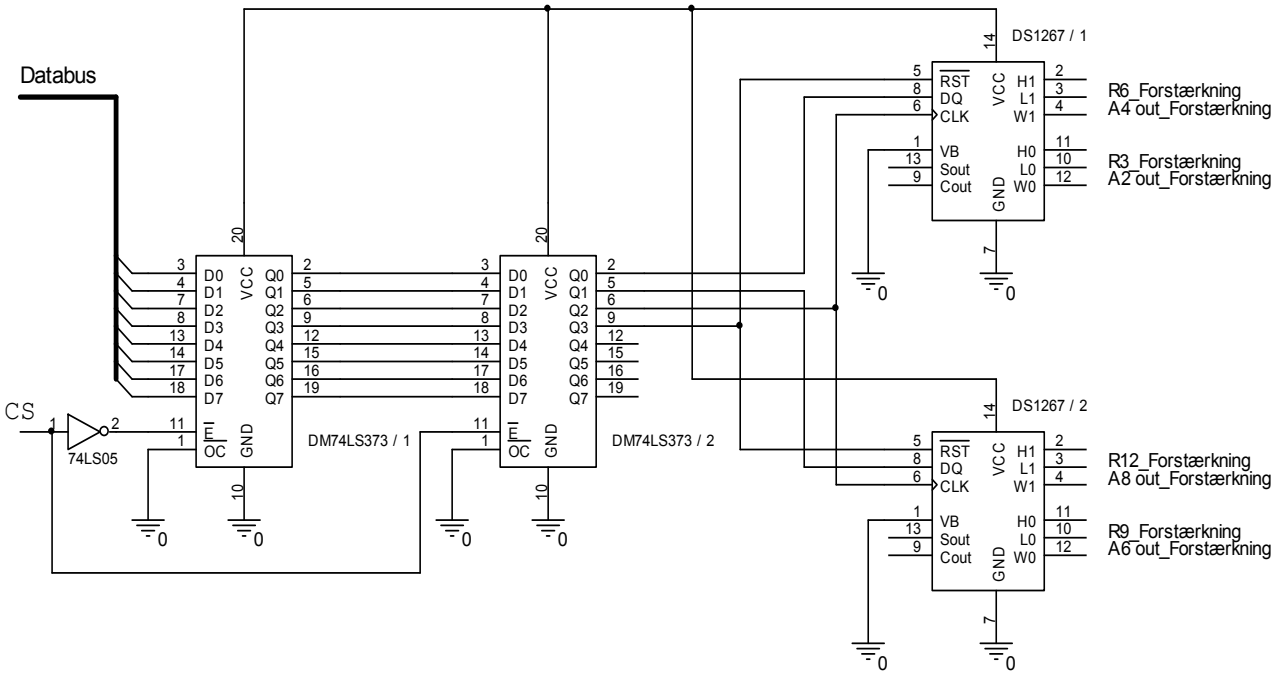
K.2 Filter



Komponent værdier:

U1-U12:	TLE2074	R5, R11		C3, C9	
		R17, R23:	240kΩ	C15, C21:	10nF
R1, R2					
R7, R8		R6, R12		C4, C10	
R13, R14		R18, R24:	478kΩ	C16, C22:	68nF
R19, R20:	7,64kΩ				
		C1, C7		C5, C6	
R3, R4		C13, C19:	122nF	C11, C12	
R9, R10				C17, C18	
R15, R16		C2, C8		C23, C24:	47nF
R21, R22:	41,2kΩ	C14, C20:	150nF		

K.3 Digitalmodstand



K.4 A/D-konvertering

Komponentværdier:

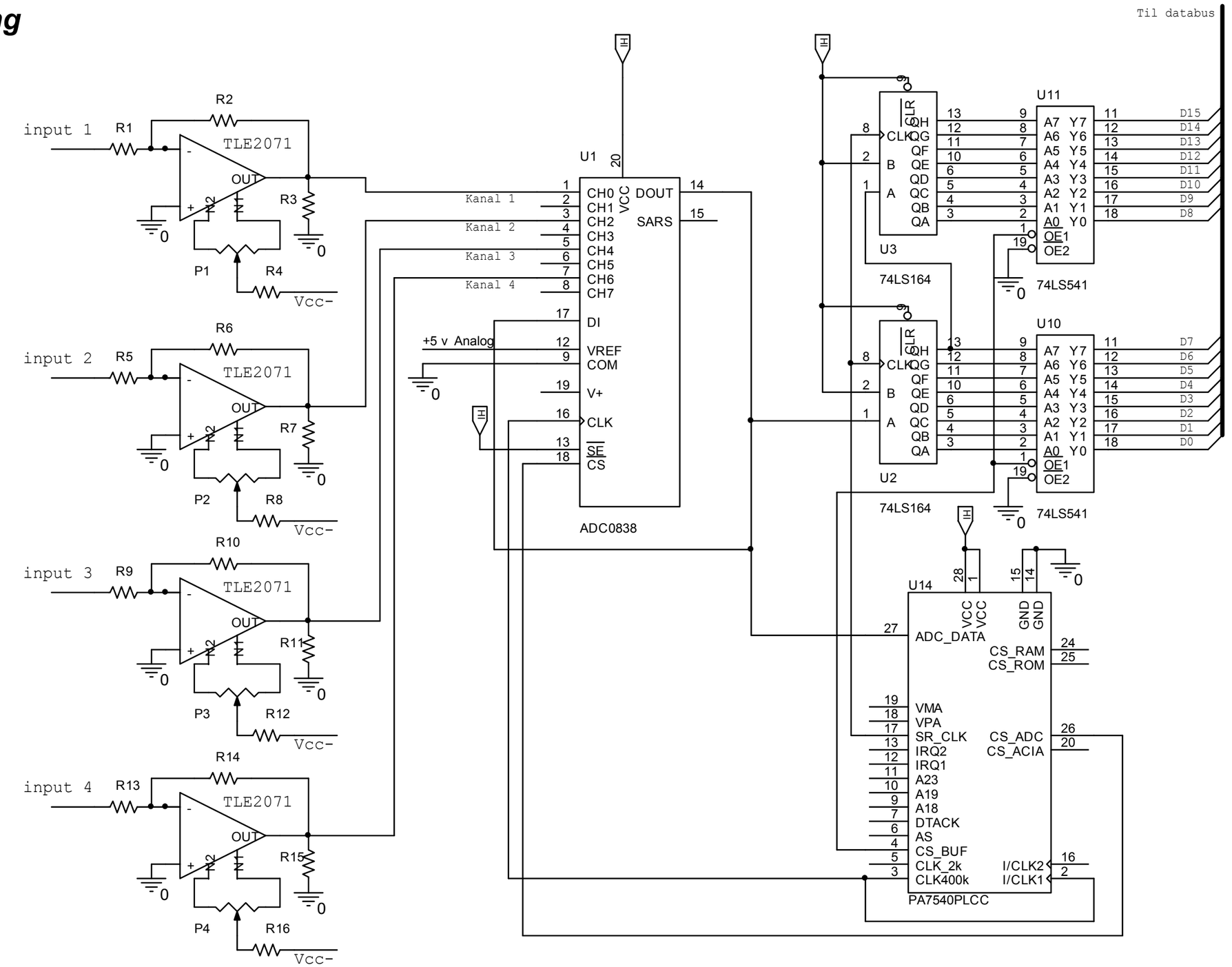
R1, R5
R9, R13: $2\,k\Omega$

R2, R6
R10, R14: $1\,k\Omega$

R3, R7
R11, R15: $1\,k\Omega$

R4, R8
R12, R16: $5\,k\Omega$

P1 – P4: $100\,k\Omega$



K.5 Mikrodatamat