Dataopsamling

-Adgangskontrol

Udarbejdet af Kenneth Dalbjerg & Ole Rosengreen, ITE21

Afleveret d. 28 maj 2003


IT og Elektronik Teknolog 2003

Erhvervsakademi
Midtjylland

Lillelundvej 29 ▪ DK-7400 Herning Telefon +45 97 12 52 00

Titel: Dataopsamling
Tema: Adgangskontrol
Projektperiode: uge 15 + 17-21 2003

2. Semester prøve

Vejledere: Heinz B. Schmidt og Benny D. Thomsen

Side antal: 154
Afsluttet: 28 Maj 2003

Udarbejdet af: Kenneth Dalbjerg & Ole Brix Rosengreen

Indholdsfortegnelse

Forord

I forbindelse med dette projekt er følgende dokumentation udarbejdet med henblik på at skabe bedre forståelse for opgaven.
Projektets tværfaglige perspektiver omfatter de forskellige fag som vi har modtaget undervisning i, i løbet af de sidste 2 semestre. Herunder IT, Elektronik og programmering, hvor vi har lagt mest vægt på programmeringsdelen.
I løbet af projektperioden har vi modtaget diverse inputs fra kollegaer og vejleder, så derfor vil vi udnytte muligheden for at takke disse.

Kenneth Dalbjerg Ole Brix Rosengreen

Formål

Formålet med dette projekt er at udvikle et adgangskontrolsystem til et given lokale. Grundlaget for problemstillingen er at øge sikkerheden og minimere brugen af de, efterhånden, gamle nøgler.
Systemet skal bestå af to programmer, et til oprettelse og redigering af brugere, og så et andet program til at styre selve login-delen.
Derudover skal der udarbejdes en fuldstændig dokumentation af ovenstående, som løbende laves gennem projektperioden.

Indledning

Adgangskontrol er i dag en væsentlig del af de fleste bygningers konstruktion, eftersom antallet af maskiner og udstyr stiger kontinuerligt sammen med værdien af disse.
Selve kontrollen kan foregå på et utal af måder, lige fra den almindelige nøgle til aflæsning af fingertryk osv.
Det betyder ikke at den almindelige nøgle har sikkerhedsfejl, men hvis man kan reducere antallet nøgler og bibeholde sikkerhedsniveauet, er man jo blevet fri for den nøgle.
Samtidig med at man udbygger adgangskontrollen, bliver der mulighed for at registrere hvem der er kommet ind og hvem der er gået, det kan man f.eks. ikke med nøglesystemet.
Vi vil anvende en magnetkortlæser som adgangsmedie, men der findes jo mange typer og varianter af kortlæsere og dertilhørende kort. For ikke at nævne de forskellige prisklasser.
Vi vælger at holde os til selve magnetkortet frem for andre kort, som smartcard, stregkodekort eller lignende. Grunden dertil er at, samtidig med vi øger sikkerheden, skal antallet af nøgler/kort reduceres.
Ved at anvende det allerede brugte studiekort, frem for f.eks. smartcard, undgår man at bære rundt på flere kort end man gør normalt og man er blevet en nøgle lettere.
Grunden til den øgede sikkerhed skyldes IKKE eventuelle eksempler på indbrud eller lignende, men derimod de ovenstående grundlag.

Projektbeskrivelse

Dette er det første udkast af projektbeskrivelsen, med de krav som vi stillede i starten af projektperioden. Vi er selvfølgelig blevet en del viden klogere i løbet af denne og har derfor lavet en del ændringer i forhold til stillede krav i løbet af perioden.
Vi har derfor, desværre, også måtte udelukke nogle af de opstillede punkter, men derimod også tilføjet nogle "features" rundt omkring , for at sætte plaster på såret.

Problemanalyse: Med den stillede opgaves kriterier, vil vi tage udgangspunkt i dette klasseværelse (117), hvor vi mener at det er muligt at forbedre adgangssikkerheden. Her på Erhvervsakademi Midtjylland er der allerede skærpet adgangskontrol på hovedindgangen, men ikke specielt på de enkelte klasseværelser, udover de udleverede nøgler. Eftersom dette klasseværelse indeholder meget elektronisk udstyr af en vis værdi, mener vi sikkerheden bør øges og at den nuværende nøgleanordning bør erstattes af en magnetkortlæser.
Som adgangskort vil vi anvende det samme kort, studiekortet, som skolen allerede anvender, for at mindske brugen af ressourcer og for at gøre det nemmere for brugeren at få adgang til lokalet.

Primær mål: Kort og godt, vil vi arbejde med Dataopsamling via en kortlæser. Til kortlæseren skal der tilsluttes et display som viser om brugeren har fået "Adgang givet" eller "Adgang nægtet".
Hele ideen er at man via kortlæseren skal kunne åbne låsen i en dør, ved et korrekt login.
Disse tre hoveddele skal forbindes til en server som opsamler og bearbejder den tilførte data.

Sekundær mål: Nu hvor vi laver en adgangskontrol til et givet rum, er det vel påkrævet at der i rummet er overvågning som giver en større

sikkerhed. Sensoren kan f.eks. være en bevægelsessensor, som ved login fra brugeren afbrydes.

Tertiær mål: At opsætte en webserver med oplysninger om hvem der er logget ind og hvor lang tid vedkommende har været der. Samtidig skal der føres log, sådan man kan se tidligere hændelser helt tilbage fra systemets opstart.

Kravspecifikation:

På serveren skal der i programmeringssproget "C" udvikles et program til styring af login via magnet-kortet. I programmet skal der være mulighed for:

1. Oprette brugere.
2. Ændre bruger/password
3. Slette brugere.
4. Føring af log over bruger-login.
   4.1 Præsentation på Web
5. Se en liste over brugere der er logget ind. Lokalt i programmet.
6. Kommunikation med serielport:
   6.1 Tænde/slukke sensorer (PIR).
   6.2 Display (Adgang givet/Adgang nægtet).
   6.3 Lås (Åbne/Lukke).

1. Ved oprettelse af brugere skal programmet opsamle et given magnetkorts unikke kode og sammenligne koden, med en anden kode som er indtastet af administratoren. Dette vil give brugeren af magnetkortet mulighed for at logge ind.

2. Hvis en bruger ønsker at skifte kort eller password, skal der i programmet være mulighed for at ændre dette.

3. Samtidig skal der være mulighed for slette allerede oprettede brugere.

4. Når brugeren så logger ind, skal programmet gemme login/logout-tidspunktet samt brugernavnet i en sideløbende tekstfil. Tekstfilen skal så senere bruges til webpræsentationen.

5. Vha. ovenstående logfil skal man også kunne, lokalt på serveren gennem programmet, kunne se en liste over de brugere der er logget ind og login/logout-tidspunktet.

6. Programmet skal ved login åbne en lås og samtidig skal to sensorer slåes fra. Ved siden af kortlæseren ved døren skal et lille display (16x4) monteres hvor det skal fremgå om alarmen er slået fra eller ej.
   Displayet skal styres af en microprocessor (PIC 16F84A) og videre styres af vores program.
   Selve kommunikationen mellem display og server skal ske gennem printerporten (LPT1).

Til serveren skal der også tilsluttes en anden kortlæser som fysisk placeres ved siden af serveren. Denne skal bruges til at tilføje brugere til systemet, eftersom afstanden mellem magnetkortlæseren ved døren og serveren, kan være meget lang. Men til vores opstilling bruger vi den samme kortlæser, fordi budgettet ikke tillader indkøb af to kortlæsere. Selve sikkerhedssystemet skal ikke bygges op efter godkendte principper da det blandt andet kræver at systemet er isoleret fra anden teknik. Og da vi har sensorerne tilsluttet serveren, har vi allerede problemer med godkendelsen.

Afgrænsning/ændringer

Som tidligere nævnt, har vi lavet en del ændringer og det fremstillede produkt lever på sin vis ikke helt op til kravspecifikationen, derfor kommer der i det følgende afsnit en kort gennemgang af mest betydende ændringer.
En meget væsentlig ting at få med, er blandt andet at vi har lavet to programmer i stedet for ét som først planlagt. Og det er der faktisk mange grunde til: Når f.eks. administratoren er ved at oprette eller redigere en bruger på serveren, kan det samme program ikke modtage signal fra kortlæseren og dvs. at brugerne ikke kan logge ind når der redigeres i bruger-strukturen.
Et program kan altså ikke administrere to læsninger fra samme magnetkortlæser, men det er løst med to programmer.
Og dog, ved at køre loginstyringen i baggrunden og brugeradministrationen i forgrunden ved CPU'en ikke hvilket program den skal sende dataene til og der opstår en forkert læsning.
Sikkerhed, en vigtig faktor som vi altså har vi udladet, pga. teoretiske overvejelser. Selve tilslutningen foregår nemlig ligesom slutblikket (låsen).
De to PIR- sendere kom ikke med selvom de var angivet som projektets sekundære mål. Og derfor har vi også valgt ikke at sikre den boks der indeholder selve den elektroniske styring.
I brugeradministrationen skriver vi at der skal være mulighed for at se en liste over de brugere der er logget ind lokalt i programmet, men nu hvor vi har lavet hjemmesiden med samme funktion, har vi udeladt den i administrations-programmet.

Kapitel

1

C Programmering Bruger Administration.

Indledning:

Vores fil til at administrere bruger er bygget op af div. funktioner, derfor består main også kun af nogle få linier.
Men det første man støder på nå man kigger kildekoden igennem, er oprettelse af en struct, det er i denne struct vi senere gemmer brugerne i.
Vi har valgt kun at gemme et brugernavn, kortnummer, kode og login status.
Bruger-informationen samt logføring gemmes i filerne "brug.txt" og "log.txt" for at de kan bruges i begge vores programmer (brugeradm. Og loginstyring).
Derudover bruges "brug.txt" til at vise hvem der er logget ind i systemet og det kan ses på hjemmesiden. Kravene til dette program fremgår af projektbeskrivelsen[1] og vil derfor ikke blive listet her.
De forskellige funktioner er overfladisk beskrevet herunder og man kan i bilag 1 se kildekoden og i bilag 2 se et principielt flowdiagram.

Vislog()
Med vislog() kan man få vist log filen, da denne log fil hurtigt bliver meget lang, viser den kun 20 linier af gangende således at man kan nå at se det på skærmen, efter den har vist 20 linier skal man trykke enter.

[1] Se Projektbeskrivelse s. 8

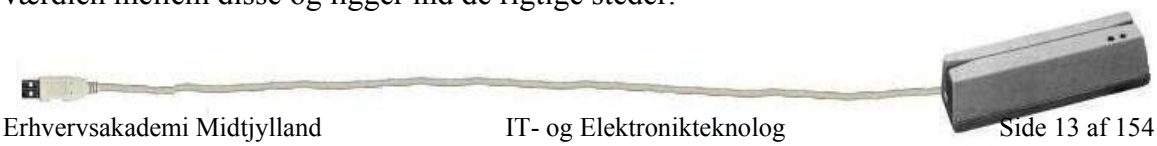
Skriv_til_logfil(beskrivelse, addlog)
Med skriv_til_logfil kan man for skrevet noget til logfilen, nå man skriver til logfilen vil linien komme til at se således ud:
Dato Klokken beskrivelse addlog. Så hvis man f.eks. har kaldt funktionen med skriv_til_logfil(”Start ”,”Program.”);
Vil den f.eks. Skrive dette til log filen:
14-05-2003 13:24:59 Start Program.

Del_bruger(bruger)
Med del_bruger slette den, den bruger som angivet som bruger.
Den gør det ved at starte for begyndelsen af structen, og så søge indtil at structen bruger er lig med bruger. Og så fjerne den dette bruger.

Opdatefil()
Opdatefil, laver en ny bruger fil, ud fra structen, altså den starte med at overskrive brug.txt (Brugerfilen), derefter gå den til begyndelsen af structen, og skriver så den ned, og gå videre til den næste, og skrive denne ned. Dette blive den ved med indtil at der ikke er flere brugere i structen. Herefter lukker den filen.

Sletbruger()
Denne funktion skal man indtaste et brugernavn, og den vil så søge efter den, via funktionen soegbruger, findes den ikke bliver man returneret til menu(), ellers vil den kalde del_bruger.

Hentbruger()
Hentbruger, tager brugerfilen (brug.txt) og åbner den til læsning. Herefter vil den slavisk gå igennem linierne, og putte værdierne ind i structen på deres rigtige pladser. Dette gør den ved at den tager linie for linier, og der brug.txt er opdelt med kolon, tager den værdien mellem disse og ligger ind de rigtige steder.



Visbruger()
Med visbruger, kan man få vist alle brugere som er oprettet. (Som er med i structen). Her gælder det igen at den ikke bare viser alle bruger, men kun nogle få ad gangen, og man skal så trykke enter.

Soegbruger(bruger)
Denne funktion søger structen i gennem, for den bruger man har skrevet sammen med funktionen. Findes brugeren vil den returnere 1, findes den ikke vil den returnere 0.

Redigerebruger()
Denne funktion bruges hvis du skal ændre en bruger, først skal man indtaste et brugernavn, her vil den så bruge soegbruger, til at se efter om brugeren findes, findes den ikke vil man blive retureneret til menu().
Findes brugeren derimod bliver du bedt om at fører et nyt kort eller det samme kort igennem kortlæseren, den læser så dette kort med funktionen laes_kort. Efter den har læst denne vil du blive bedt om at taste en ny kode ind, til brugeren.
Den gør så dette at den først slette brugeren via del_bruger, opdatere bruger filen via opdatefil og så opretter den en ny bruger, ved hjælp af opretbruger.

Laeskort()
Læser ude på com port1, eller adresse 0x3F8, og det bliver den ved med indtil den får et ? sendt fra com porten. (Alle magnet kort slutter med en ?). Så den vil stoppe lige efter man ført et magnetkort rigtigt igennem. Koden på magnet striben bliver så returneret til der, hvor funktionen laeskort, blev kaldt.

Opret()
Brugeres til at oprette en bruger, her vil man blive bedt om at indtaste brugernavn, derefter skal man fører kortet igennem. (Kortet bliver så læst via laes_kort(), og til sidst skal man skrive brugerens kode. Den opretter så brugeren ved hjælp af en anden funktion opretbruger. Dog tjekker den efter om man allerede har lavet en bruger med samme navn.

Opretbruger(VAnavn, VAkortnummer, VAkode)
Åbner brug.txt for skrivning, hvorefter den putter Vanavn, Vakortnummer, Vakode i med kolon i mellem, og så login status. Den opretter den så også i structen.

Credits()
Fortæller hvem som har programmeret systemet og hvornår.

Menu()
Viser menuen, og kalder den funktion man nu har valgt i menuen.

Main()
Gør kun 2 ting, skriver til log filen at programmeret er startede og kalder så menu().

Delkonklusion:

Som nævnt i afgrænsningen har vi bygget systemet op i to programmer i stedet for ét, fordi det ikke kunne lade sig gøre at oprette brugere og køre login-styring på samme tid. Og da vi kun har én kortlæser, kan vores program stadig ikke oprette brugere og køre login-styring samtidig, eller dvs. praktisk set så kan man godt, men det skaber konflikter mellem de to programmer.
Konflikterne vises når man vil oprette en bruger, altså læse fra COM1, samtidig med at login-styringen læser på COM1. Så ved UART'en ikke hvilket program den skal sende dataene til og der kommer en forkert læsning af magnetkortet.
Det optimale ville være hvis vi havde to kortlæsere, sådan vi kunne læse fra to serielle porte samtidig og derved køre begge programmer samtidig, men prisen er ikke helt billig.
Ellers lever programmet op til de krav der er stillet i projektbeskrivelsen, med lidt skønhedsændringer undervejs.

Kapitel

2

Login-styring

Indledning:

Efter at brugeren er oprettet via programmet "Brugeradministration", i forrige kapitel, er han klar til at anvende selve loginprogrammet[2]. Man skal være oprettet med magnetkort og kode for at kunne anvende systemet, hvilket jo også er meningen med det hele.
Programmet er bygget op omkring tekstfilen "brug.txt" som indeholder alle informationer om de forskellige brugere.
Filen indeholder brugernavn magnetkortnummer, adgangskode og loginstatus.
Brugernavnet er i selve login-styringen irrelevant og bruges kun i log-føringen og i brugeradministrationen, for nemmere at kunne holde styr på de forskellige brugere.
Bilag 4 er et flowdiagram over programmets virkemåde og kan med fordel anvendes til at danne sig et billede over programmet som helhed.

Programbeskrivelse:

Det første programmet gør, er at opsætte I/O-kortet.[3] Dernæst skrives der til logfilen[4] at programmet er startet.
Herefter kaldes funktionen start() der indeholder de forskellige underfunktioner som styrer selve programmet.
Det første kald i start() er hentbruger() som opdaterer den struct der indeholder brugerne, kortnummer, loginstatus osv. Formålet er at finde ud af om der er logget nogen ind eller ej, men det bliver først testet i den næste funktion antallog().

[2] Se kildekode i bilag 3.
[3] Se kap 4 side 26
[4] Se kap 1 side 13

Antallog()

Når den første bruger kommer for at logge ind, vil døren være låst og skal derfor åbnes. Døren låses først når den sidste bruger logger ud.

Funktionen antallog(), bruges som sagt til at holde øje med, om der er logget nogen ind. Den læser structen igennem og hvis alle log-statusser er nul, vil døren låses op ved login og låses ved logout.

Når brugerstructen er opdateret sammen med loginstatusen, vil der på displayet blive skrevet "Indlaes kort/tast kode!!" . Funktionen laes_kort() bliver herefter kaldt og programmet er klar til brug.

Laeskort() og Laeskode()

Det er i denne funktion, laes_kort(), at hele essensen er, men den er beskrevet i et andet afsnit der omhandler kortlæseren.[5] Det er også i laeskort() at funktionen laeskode() bliver kaldt og det er i denne at koden bliver indtastet og gemt i en streng. Også laeskode() bliver beskrevet i et andet kapitel, matrix-tastatur[6].

Login_out()

Når kortet og koden er accepteret, skal det registreres at der er en bruger som enten er logget ind eller logget ud. Det gøres vha. funktionen login_out() der kaldes med det indlæste kortnummer og hvis det bliver fundet i structen, vil kortnummeret bliver overført til funktionen opdatefil().

Opdatefil()

Kort sagt, ændrer den login-statusen. Dvs. at når man logger korrekt ind, vil ens status blive ændret fra "0" til "1" i "brug.txt" og omvendt. Det sker ved at løbe hele structen igennem fra start indtil man så i structen møder det kortnummer der matcher det læste

[5] Se kap 3 side ????
[6] Se kap 7 side ????

kortnummer. Her tjekkes der så efter om brugeren er logget ind, er han det, skal der skrives ”0” og er han ikke skal der skrives ”1”. Herefter fortsættes der indtil structen er kørt helt igennem.
Når denne funktion er udført, returneres der til funktionen start() og programmet starter forfra.

Delkonklusion:
Loginstyringen består af mange forskellige funktioner der styrer eksterne enheder og selvfølgelig styring af selve login/logout princippet. Derfor faldt det os mest naturligt at forklare de forskellige enheders (kortlæser, tastatur mm.) c-kode i de kapitler som omhandler enhederne. F.eks. c-koden til tastaturet er beskrevet i kap 7, som omhandler tastaturet. Derfor har vi lavet denne lille liste over funktionerne i programmet og de kapitler hvori de er beskrevet.

Funktionsoversigt:

Funktion:	**Kapitel:**	**Emne:**
Delay()	Kap 4	I/O kort
Antallog()	*Kap 2*	*Loginstyring*
skriv_til_log()	Kap 1	Brugeradministration
start()	*Kap 2*	*Loginstyring*
hentbruger()	Kap 1	Brugeradministration
laeskode()	Kap 7	Numerisk tastatur
laeskort()	Kap 3	Kortlæser
login_out()	*Kap 2*	*Loginstyring*
opdatefil()	*Kap 2*	*Loginstyring*
dioder()	Kap 4	I/O kort
display()	Kap 6	Programmering med PIC16C84A
main()	*Kap 2*	*Loginstyring*

NB. De kapitler som er markeret med gråt er beskrevet i dette kapitel.

Kapitel

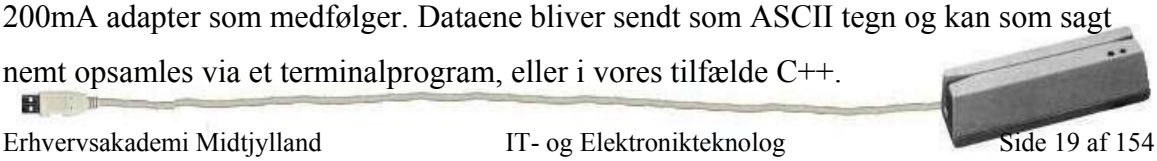
3

MCR12S Magnetkortlæser

Der findes mange forskellige kort, såsom kort med chip, kort med en magnet og f.eks. kort med stregkode. Vi har valgt kort med en magnet, på grund af at vi gerne vil undgå at man skulle have flere kort og nøgler. For at kunne bruge vores studiekort, så skal det være magnetkortlæser.
Smartcard, er en af dem med en chip på. Chipkort er meget mere sikker end f.eks. Magnetkort, da man på en chip kan kryptere indholdet på kortet, og det derved er svære at kopiere et sådan kort.
Magnet kort, er dem de fleste bruger i dag, og blandt andet derfor vi har valgt dette. Da vi ikke ønsker at give brugeren flere kort, men blot at han skulle kunne benytte nogen af dem han allerede har.

Teknisk specifikation:

Magnet kort har 3 uafhængige sport, Spor 1: Alfanumerisk, 79 tegn. Spor 2: Numerisk, 40 tegn. Spor 3: Numerisk, 107 tegn. Og alt efter hvilken skriver / læser du har kan du bruge nogen af dem, eller alle sammen.
Vores læser kan læse spor 1 og spor 2. For at få det sidste spor med, bliver læserne ca. dobbel så dyre (danbit priser).
Vores erfaringer siger os at alle kort er bygget op ens når det angår den unikke kode på magnetstriben, dvs. at det først starter med et ; og slutter med et ?. Og ind i mellem kan da være tal og bogstaver, alt efter hvilket spor man har gemt det på.
MCR12S Tilsluttes RS232 (serielporten). Software er påkrævet for at aflæse data fra enheden. Funktionen kan afprøves med et terminalprogram (PROCOMM, TELIX m.fl.). Data sendes med 9600 baud, ingen paritet, 8 bit og 1 stopbit. Strømforsynes af 5VDC 200mA adapter som medfølger. Dataene bliver sendt som ASCII tegn og kan som sagt nemt opsamles via et terminalprogram, eller i vores tilfælde C++.

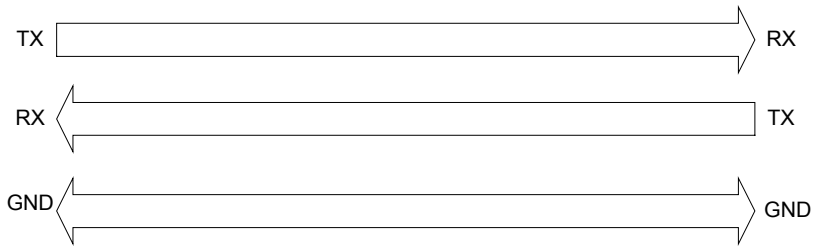


Virkemåde:

Kortlæseren virker på den måde at nå man fører et kort igennem, kommer der straks en strøm af data, som ASCII til serielporten. Kortlæseren gemmer altså ikke disse data i hukommelsen og venter med at sende dem, til man spørger. Den sender bare og er man ikke klar til at modtage, gå der dataene tabt. Dette skal man så lave sådan at programmet laver en fejl meddelelse. Det gør vi ved at der lyser en rød diode, og der i displayet står fejl.

Seriel kommunikation:

foregå over 2 ledere, hvis der skal data begge veje. Man kalder normalt den leder som aflevere data for RX og den leder som sender for TX, (Recieve / Transmit). Når data skal over længere afstand bruger som regel altid Seriel kommunikation, fordi at der kun skal 2 ledere til. Hvor i mod med parallelt kommunikation skal der flere til. Selv om man bruger 2 ledere til at sende data, bruger man dog tit en ekstra for at give de kommunikerende enheder, en fælles stel. Som vist herunder:



Disse tre ledere er faktisk nok til at kunne lave en seriel kommunikation mellem 2 computere eller en anden seriel enhed som f.eks. vores kortlæser.
Da man selvfølgelig har sat begge computer til at køre med samme speed, er flow control ikke nødvendigt.

Flow Control:

Hvis man sætter et modem og en pc sammen, og sætter computeren til at sende hurtigere end modemet kan sende videre, vil modems buffer før eller siden blive fyldt op. Og så har modemet brug for at fortælle computeren at den skal stoppe med at sende. Flow control har både en software og en hardware variant.
Software Flow Control, bruger 2 Karakterer, en for Xon og en for Xoff. Dette er normalt ASCII 17 & ASCII 19 karakter Hvor 17 er for Xon og 19 er for Xoff. Når så modemets, buffer er fyldt, vil den sende et Xoff tegn af sted til computeren for at fortælle at den skal stoppe med at sende data. Nå den så er klar igen, vil den sende et Xon tegn. Dette tager 10 bit, og kan betyde at på langsomme forbindelser, vil det sænke kommunikationshastigheden.
Hardware Flow Control, bruger 2 ekstra ledere I stedet for 2 tegn, og derved sløver den heller ikke kommunikations hastigheden ned. Nå så computeren vil sende data til modemet, bliver computer aktiv på Request to Send. Hvis så modemet er klar til at modtage data, vil den svare tilbage på Clear to Send, hvis det der i mod ikke er klar til at modtage data vil den ikke sende en Clear to Send.
Dog bruger kortlæseren ikke flow kontrol, da den blot sender til computeren, og altså ikke modtager noget som den skal gemme.

Paritetsbit:
En paritetsbit er en bit, som sættes i forlængelse af en byte eller som ottende bit i en byte og benyttes som paritetscheck. Altså noget som tjekker om computeren har modtaget et lige antal 1 eller et ulige antal 1.

Programmering:
I starten af programmet har vi defineret # define PORT1 til 0x3F8. Dette giver blot PORT1 en værdi som er 0x3F8. Så man bare fremover skal skrive PORT1, i stedet for det tal. Denne værdi er adressen på COM1.
Først skal porten initialiseres, med hvilket hastigheder, paritet bit, osv.

```
Outportb(PORT1 + 1 , 0);/* Turn off interrupts - Port1 */
```

Der sendes ikke et interrupt til CPU'en når FIFO/receive registeret indeholder data.

```
Outportb(PORT1 + 3 , 0x80);  /* SET DLAB ON */
```

DLAB (Divisor Latch Access Bit) bruges til at lave 8 port-adresser om til 12 registre. Det gøres ved at DLAB sættes til "1" eller "0" via linecontrol registeret og derved fremkommer der to ekstra registre hvori man f.eks. kan sætte baudraten.

```
Outportb(PORT1 + 3 , 0x03);/* 8 Bits, No Parity, 1 Stop Bit
*/
```

Porten sættes til at modtage 8 bit uden paritets-tjek og så en enkelt stopbit.

```
Outportb(PORT1 + 0 , 0x0C);  /* Set Baud rate - Divisor
Latch Low Byte */
Outportb(PORT1 + 1 , 0x00);  /* Set Baud rate - Divisor
Latch High Byte */
```

Som man kan se herunder, sættes baudraten til 9600 BPS med de to ovenstående kald.

Speed (BPS)	Divisor (Dec)	Divisor Latch High Byte	Divisor Latch Low Byte
50	2304	09h	00h
300	384	01h	80h
600	192	00h	C0h
2400	48	00h	30h
4800	24	00h	18h
9600	12	00h	0Ch
19200	6	00h	06h
38400	3	00h	03h
57600	2	00h	02h
115200	1	00h	01h

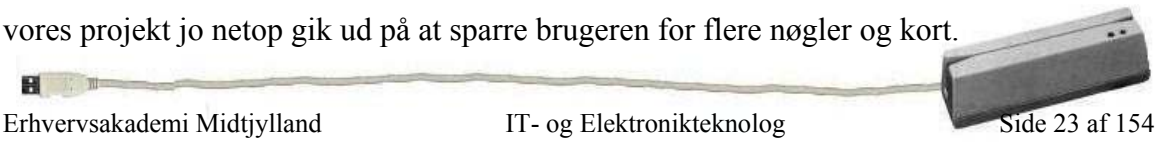
```
outportb(PORT1 + 2 , 0xC7);  /* FIFO Control Register */
```

FIFO Control registeret er det register som anvendes til at opbevare den første bit der bliver sendt fra kortlæseren. FIFO = First In/First out.
Herefter skal vi vente på data fra kortlæseren, dette gør vi en do while løkke, hvor den bliver ved med at opsamle "karakterer" indtil at der kommer et ? tegn fra kort læseren. (Som skrevet tidligere slutter alle magnetkort med et ?.)
Vi har så også lige lavet en esc funktion her. For at man kan afslutte programmet hvis noget skulle gå galt.

```
do {
     c = inportb(PORT1 + 5);
Her læser vi port-statusen og hvis der kommer et signal
bliver c = 1 og if-sætningen nedenunder bliver sand.
     if (c & 1) {
          ch = inportb(PORT1);
          s_temp[j++] = ch;
     }
     if(kbhit()) {
          tegn = getch();
     }
     if(tegn == 27) {
          exit(0);
     }
}
while (ch !='?');
```

Delkonklusion:

Der vi først havde en smartcar reader, som kørte på USB, havde vi lidt problemer med at vores projekt jo netop gik ud på at sparre brugeren for flere nøgler og kort.



Da den så også kørte USB, var det mere svære at får lavet noget C++, som gad at arbejde sammen med den. Vi fik dog fat i en magnetkort læser som kørte med et RS232 stik, og så var det lige til.
Dog havde vi lidt problemer i starten med denne, men det var fordi vi havde sat vores port op til at læse for hurtigt for kortlæseren og der kom så bare nogle underlige tegn, fordi magnetkortlæseren ikke kunne sende så hurtigt.

Kapitel

4

I/O –kort ACL-7124

Indledning

I/O channels	24
Input Signal	Logic High Voltage :2.0 V to 5.25V Logic Low Voltage : 0.0 V to 0.80V Logic High Current : 20.0 uA Logic Low Current : -0.2 mA
Output Signal	Logic High Voltage : Minimum 2.4 V Logic Low Voltage : Maximum 0.5V Logic High Current : -15.0 mA Logic Low Current : 24.0 mA
Operating Temperature	0° ~ 60° C
Storage Temperature	-20° ~ 80° C
Humidity	5% ~ 95% non-condensing
I/O Connector	50-pin male ribbon cable connector
Bus	PC/XT Bus
IRQ Level	IRQ2 ~ IRQ7
I/O port address	4 bytes(Hex 200 ~ Hex 3FF)
Power Consumption	0.5A @5VDC (Typical) 0.8A @5VDC (Maximum)
Transfer Rate	300 K bytes/sec (Typical) 500 K bytes/sec (Maximum)
Size	(119mm X 97mm)

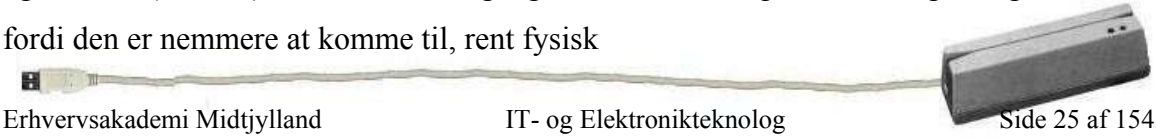
Kortet er udstyret med 24 digitale I/O – porte og tilsluttes computeren via ISA-bussen.

De 24 porte er delt op i tre grupper PA, PB og PC, som er henholdsvis Port A,B og C, hvor port A og B er fast 8 bit og C kan indstilles til både at køre 8 og 4 bit.

Når man skal skrive til kortet er det vigtigt at kende dets adresse. Det er BASE adressen der her refereres til og er som standard sat til 0x2C0, men kan ændres vha. den DIB-switch der sidder på kortet. BASE adressen kan være alt mellem 0x200 og 0x3FF, men må naturligvis ikke konflikte med computerens adresser. Bag i manualen til I/O-kortet er der en oversigt over de adresser som allerede er anvendt af computeren og man kan ud fra dette appendiks definere sin egen, hvis det skulle være nødvendigt.

Derudover er der mulighed for at ændre kortets IRQ hvis det nu skulle konflikte og det gøres vha. en jumper. Der kan her vælges mellem IRQ'er fra 2-7 og standard er 2.

Man har adgang til de 24 porte gennem et 25 pins (CN1) stik bag på kortet, men der er også 2 stik (CN2+3) hvor man har adgang til de samme 24 porte. Vi bruger dog kun CN1 fordi den er nemmere at komme til, rent fysisk



Til højre ses en tabel over kortets tekniske specifikationer, men disse kan også ses i databladet i bilag 12.

C-kode til I/O-kort

Inden man begynder at skrive til de forskellige porte, skal de først defineres, altså initialiseres til enten output eller input.

BASE + 0 = Port A (PA)
BASE + 1 = Port B (PB)
BASE + 2 = Port C (PC)
BASE + 3 = CWD (Control Word) = Opsætning Input/output.

Her til højre er en oversigt over de forskellige kombinationer af input/output, hvor man i venstre kolonne kan se HEX koden. 90H er den værdi vi bruger og som laver Port A til input og Port B/C til input.

Config værdi	D4	D3	D2	D1	Port A	Port C upper	Port B	Port C lower
80H	0	0	0	0	O/P	O/P	O/P	O/P
81H	0	0	0	1	O/P	O/P	O/P	I/P
82H	0	0	1	0	O/P	O/P	I/P	O/P
83H	0	0	1	1	O/P	O/P	I/P	I/P
88H	0	1	0	0	O/P	I/P	O/P	O/P
89H	0	1	0	1	O/P	I/P	O/P	I/P
8AH	0	1	1	0	O/P	I/P	I/P	O/P
8BH	0	1	1	1	O/P	I/P	I/P	I/P
90H	1	0	0	0	I/P	O/P	O/P	O/P
91H	1	0	0	1	I/P	O/P	O/P	I/P
92H	1	0	1	0	I/P	O/P	I/P	O/P
93H	1	0	1	1	I/P	O/P	I/P	I/P
98H	1	1	0	0	I/P	I/P	O/P	O/P
99H	1	1	0	1	I/P	I/P	O/P	I/P
9AH	1	1	1	0	I/P	I/P	I/P	O/P
9BH	1	1	1	1	I/P	I/P	I/P	I/P

Kommandoen sendes til I/O-kortet vha. Outportb(adresse, kommando) og det er opsat som vist herunder:

```
outportb(BASE + 3, CWD);
```

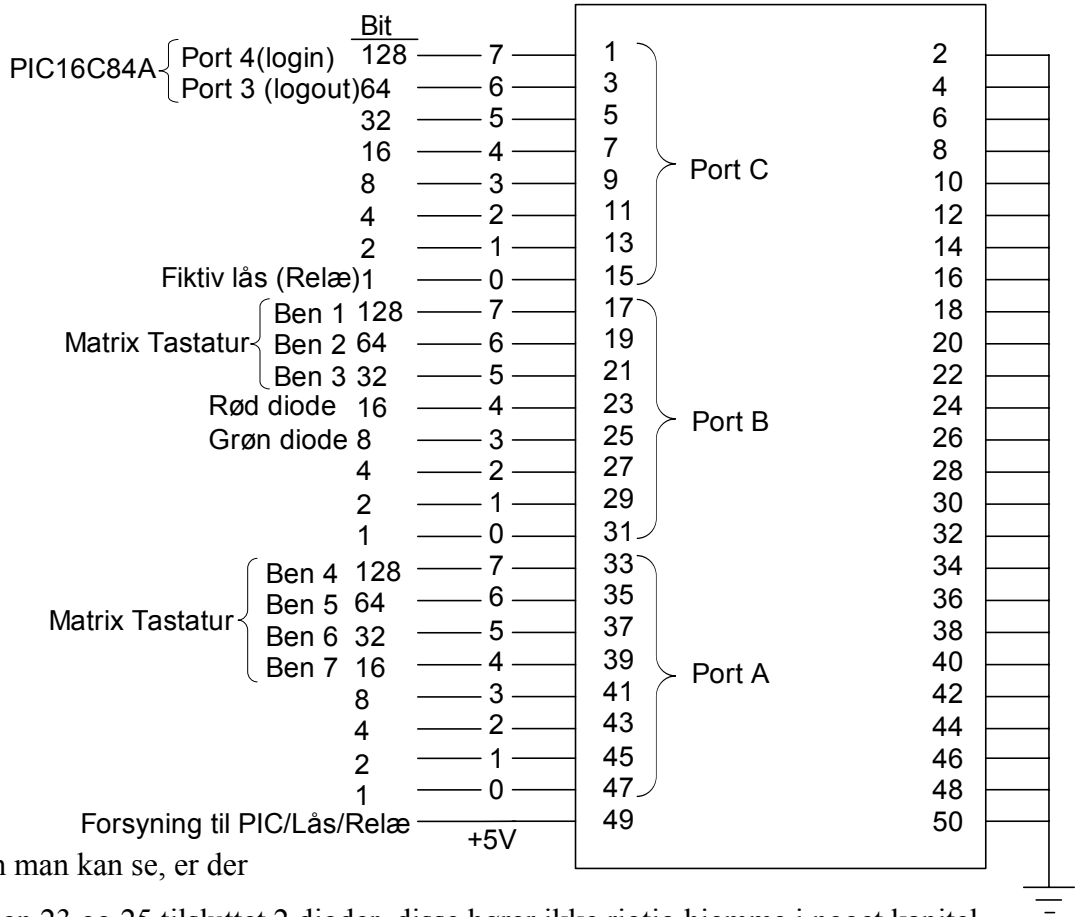
Da vi kun anvender denne opsætning, bliver ovenstående initialiseret i main() og ikke andre steder. De forskellige definitioner af portene A-C samt CWD bliver indlæst i starten af "login.c" [7].

[7] Se bilag 3, side 1.

```
# define BASE 0x2C0 // define base address
# define PA 0x0 // Port A
# define PB 0x1 // Port B
# define PC 0x2 // Port C
# define CWD 0x90 //PORT A INPUT, PORT B/C OUTPUT
```

Al den C-kode som er skrevet med det formål at kommunikere med I/O-kortet er godt spredt ud over hele den samlede kode. Derfor har vi valgt at gennemgå kildekoden sammen med de enheder som anvender det. Men generelt kan man sige at kommunikationen til de forskellige porte er ens. Hver port har en Binær-værdi som ved kaldet (Port, bit) kan aktiveres. Eks. outportb(BASE + 2, 128); vil aktivere ben 1 på I/O-kortet.

Herunder er en oversigt over alle de anvendte porte på I/O-kortet og forbindelserne til de enkelte enheder:



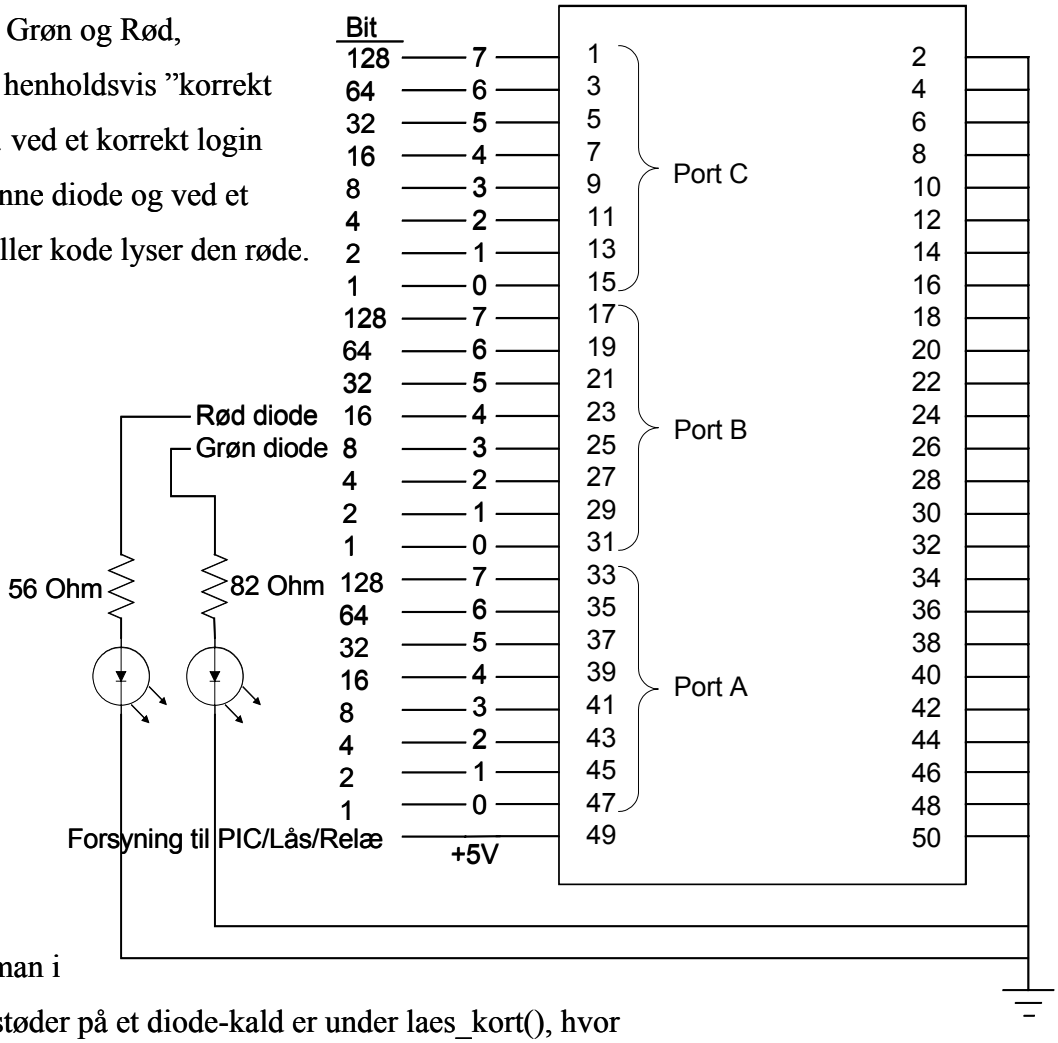
Som man kan se, er der

på ben 23 og 25 tilsluttet 2 dioder, disse hører ikke rigtig hjemme i noget kapitel, så derfor kommer der en kort forklaring i det følgende afsnit.

Alle enhederne kører på 5V forsyning og det trækkes direkte fra I/O-kortets ben 49 og alle stel-forbindelserne er tilsluttet et ben fra 2-50.

Dioderne:

De to dioder, Grøn og Rød, symboliserer henholdsvis ”korrekt og fejl”. Dvs. ved et korrekt login lyser den grønne diode og ved et forkert kort eller kode lyser den røde.



Første gang man i kildekoden[8] støder på et diode-kald er under laes_kort(), hvor den grønne diode, kort skal indikere at der er ført et korrekt magnetkort gennem kortlæseren.

Selve funktionen dioder() består af to if-sætninger, som hver kalder port B med en forskellig binær-værdi. Hvis b’00010000’ sendes lyser den grønne og hvis b’00001000’ sendes, vil den røde lyse.

Ellers styres dioderne sammen med funktionen display(), fordi de skal virke synkront sammen med displayet.

[8] Se bilag 3 side 89

Hver ben på I/O -kortet kan supplere en spænding på 3,6 V, der ligger sig 2V[9] over dioden. Dette minuser man med hinanden, og dividere så tallet med diodens minimums strøm, for at lyse. Som for rød er 30mA[10] .

$\frac{1.6}{30} = 53 \approx 56\Omega$ Jævnfør E12 rækken.

For grøn er tallet minimums strømmen dog kun 20mA. Så her vil det blive

$\frac{1.6}{20} = 80 \approx 82\Omega$ Jævnfør E12 rækken.

For at være sikker på at de modstande vi har brugt ikke brænder af, så vil vi lige regne effekten som vil blive afsat i modstanden ud. Dette gøres med P = U * I.
Da vi har 3.6 Volt forsygning, og der dioden tager de 2V, vil der ligger sig 1.6 volt over modstanden. Og der vil gå 30mA for en rød lysdiode, og 20mA for en grøn lysdiode. Derfor vil effekten for en rød lysdiode blive

$\frac{1,6}{30} = 53,33mW$

og for en grøn lysdiode vil dette blive

$\frac{1,6}{20} = 80mW$

Dette har dog ingen betydning, da modstanden vi har brugt kan holde til 250mW.

Resumé:

I/O-kortet bruges til at styre alle de eksterne enheder undtagen magnetkortlæseren. Kortet har 3 forskellige porte A, B og C der virker uafhængigt af hinanden, til vores fordel. En oversigt over hvilke enheder der er tilsluttet hvor kan ses på side 25.

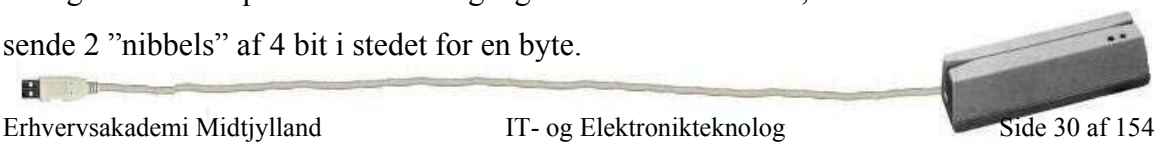
[9] Se bilag 15 side 139
[10] Se bilag 15 side 140

Kapitel

5

20x4 Character Type Dot Matrix LCD module

Også kaldt ”Liquid Crystal Display” eller bare LCD. Vi har anvendt dette display til at vise brugeren om han f.eks. har tastet adgangskoden korrekt, eller om han i det hele taget bruger det rigtige magnetkort.
20x4 betyder at der i displayet er plads til 20 bogstaver horisontalt og 4 bogstaver vertikalt, eller
at det er 20 tegn bred og 4 tegn højt.
Producenten er den kinesiske LCD-fabrikant Jinghua, men det er selvfølgelig købt hos en underforhandler her i landet. Den helt præcise forhandler kan vi ikke oplyse, eftersom displayet er en efterladenskab fra en af de tidligere års, studerende. Vi har søgt hos forskellige forhandlere og det lader umiddelbart til at produktet er udgået, så derfor er prisen også ukendt.
Vi har dog fundet frem til at displayets modelnummer er JM204A og at kontrolleren er KS0070B.
Jævnfør databladet for displayet, skulle den kontroller svare til en kontroller fra Samsung, S6A0700, som så igen svarer til en standardiseret kontroller, Hitachi HD44780.
Med disse informationer har man nu mulighed for at finde diverse testprogrammer, til at prøve om displayet overhovedet virker!
Vi startede med at ville styre displayet parallelt gennem LPT1 og så sende instruktionerne via programmeringssproget C. Men trods mange ihærdige forsøg kunne vi overhovedet ikke skabe forbindelse til displayet, trods det at vores display var baseret på HD44780. Så derfor valgte vi at anvende en mikroprocessor i stedet for, men det vender vi tilbage til i et senere kapitel. Og det er her at HD44780 virkelig viser sit værd, den er nemlig bygget til at være kompatibel med 4-bit mikroprocessorer. Bla. PIC-processorerne. Men vi har jo heldigvis en mikroprocessor med udgange nok til at køre 8-bit, sådan at vi ikke skal sende 2 ”nibbels” af 4 bit i stedet for en byte.



For at kunne styre alle 4 linier er displayet udstyret med en 8 bit databus, som man har adgang til på benene 7-15. Derudover er der et ben til enable/disable, ben 6, og et ben til Read/write, ben 5, og til sidst et ben til "register select", ben 4.

Den samlede oversigt over displayets 16 ben, ses herunder:

1	GND	Power supply	0 V
2	Vdd		+5 V
3	V5		For LCD
4	RS	Register Select (H=Data, L=instruction)	
5	R/W	Read/write	
6	E	Enable	
7	DB0	Data bus bit 0	
8	DB1	Data bus bit 1	
9	DB2	Data bus bit 2	
10	DB3	Data bus bit 3	
11	DB4	Data bus bit 4	
12	DB5	Data bus bit 5	
13	DB6	Data bus bit 6	
14	DB7	Data bus bit 7	
15	A	Anode of LED Unit	
16	K	Cathode of LED Unit	

Som man nok bemærker, hvis man kigger på vores opstilling af displayet, er de sidste to ben(15,16) ikke tilsluttet. Grunden dertil er, at de kun bruges til displays med baggrundsbelysning og det har vores ikke. Grunden til at benene stadig er der, skyldes igen standardiseringer, hvilket jo bare gør det hele nemmere for os, som brugere.
Som sagt er displayet opbygget af 20x4 karakterer, men hver eneste karakter eller tern, er bygget op af dots. Hver karakter består af 5x7 dots, hvilket er rart at vide hvis man vil generere sine egne brugerdefinerede tegn.

Displayet har to væsentlige RAM-blokke, nemlig *CGRAM* og *DDRAM*.

CGRAM:

Denne kan bruges til de førnævnte brugerdefinerede tegn, hvorved man manuelt sætter hver eneste dot og derfter gemmer det i CGRAM'en. Hver gang displayet reset'es, reset'es denne hukommelse også, så derfor bør tegnene initialiseres ved opstart af kontrolleren, som i dette tilfælde er PIC-processoren.

CGRAM står for "Character Generator RAM" og udgør hele 64x8 bits eller 8 karakterer i størrelsen 5x7.

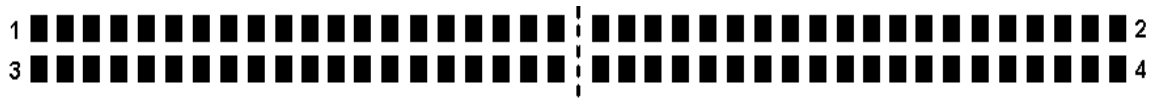
DDRAM:

"Display Data RAM" bruges til, midlertidigt at opbevare de tilførte karakterer. Derved skal man ikke skrive til displayet hele tiden for at få en konstant tekst-streng vist. Her er der så lidt mere hukommelse, 80x8 bits eller 80 karakterer, hvilket gør det muligt at gemme karakterer til hele displayet.

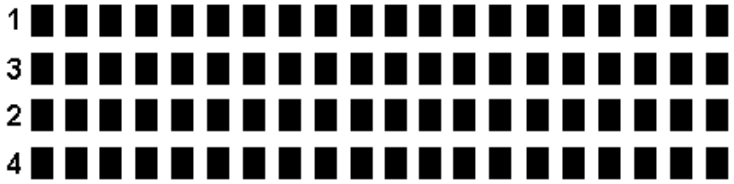
Faktisk indeholder displayet allerede definerede karakterer som er gemt i CGROM'en, der har en hukommelse på 8320 bits eller 192 karakterer. Det kaldes også ASCII tegntabellen og er meget nyttefuld, nu hvor det er den vi skal bruge når vi skriver tekst ud på displayet.

2 eller 4 linier?

Selvom, det af navnet ”20x4” fremgår, at displayets visuelle udformning har 4 linier, opfatter kontrolleren det på en anden måde. Den ser det nemlig som 2 linier. Umiddelbart giver det ikke meget mening, men det skulle den nedenstående figur gerne hjælpe lidt på.



Kontrolleren opfatter faktisk displayet som et 40x2 og det kan også ses hvis man fylder hele linie 1 med tegn og så tilføjer et ekstra tegn. Dette ekstra tegn ville vi så gerne have vist på linie 2, men det kommer på linie 3, jævnfør kontrollerens instruktioner. Displayets linier er som vist her til højre:



Initialisering af display

Uden den korrekte opstart eller initialisering kan displayet ikke vise nogen form for bogstaver eller karakterer. Derfor er der, som sagt, tre ben til styring af dette. (Ben 4,5,6). Det første ben, 4, er som før nævnt ”register select”. Når denne port er lav, opfatter displayet alle de tilførte data som kommandoer eller instruktioner til opsætning. Samtidig vil en læsning af displayet i denne tilstand angive displayets status. F.eks. ”busy”. Busy-time er den tid det tager at udføre instruktionerne og i den tid kan der ikke laves andet, men det vender vi tilbage til senere.
Når den så sættes høj bliver der mulighed for at sende karakterer til og fra displayet. Den næste, ben 5 (read/write), vil i lav tilstand give brugeren mulighed for at skrive, altså sende karakterer til displayet. Ved at sætte denne port høj kan man aflæse status og karakterer fra de førnævnte registre.

Ben 6, det sidste i rækken, er ”enable” og er displayets on/off funktion. Dvs. at denne funktion styrer om der skal skrives eller læses fra displayet, lidt ligesom ”ben 5”. Men overførslen af kommandoer eller karakterer sker kun når der skiftes fra høj til lav på dette ben. Derimod kan læsning fra displayet ske i hele den periode hvor signalet er høj.
Når man kender funktionen af disse ben, kan man starte på den grundlæggende opsætning af bla. ”Cursor”.
Det nedenstående skema giver et lille indtryk af hvilke indstillinger man skal tage stilling til inden man begynder den egentlige overførsel af tekst til displayet.

Command	Binary								Hex
	D7	D6	D5	D4	D3	D2	D1	D0	
Clear Display	0	0	0	0	0	0	0	1	01
Display & Cursor Home	0	0	0	0	0	0	1	x	02 or 03
Character Entry Mode	0	0	0	0	0	1	1 / D	S	04 to 07
Display On/Off & Cursor	0	0	0	0	1	D	U	B	08 to 0F
Display/Cursor Shift	0	0	0	1	D / C	R / L	x	x	10 to 1F
Function Set	0	0	1	8 / 4	2 / 1	10 / 7	x	x	20 to 3F
Set CGRAM Address	0	1	A	A	A	A	A	A	40 to 7F
Set Display Address	1	A	A	A	A	A	A	A	80 to FF

1 / D:	1=Increment*, 0=Decrement	R / L:	1=Right shift, 0=Left shift
S:	1=Display shift on, 0=Off*	8 / 4:	1=8-bit interface*, 0=4-bit interface
D:	1=Display on, 0=Off*	2 / 1:	1=2 line mode, 0=1 line mode*
U:	1=Cursor underline on, 0=Off*	10 / 7:	1=5x10 dot format, 0=5x7 dot format*
B:	1=Cursor blink on, 0=Off*		
D / C:	1=Display shift, 0=Cursor move	x = Don't care	* = Initialization settings

Figur 1:

En indstilling kunne f.eks. være 8 bit interface, med 2 linier og 5 x 7 dot format. Den binære værdi vil så være b’00111000’ eller i HEX 0x38. Selv samme indstilling bruger vi som vores opsætning.
En vigtig ting at tage hensyn til er, at displayet ikke arbejder ligeså hurtigt som f.eks. den PIC-processor vi arbejder med. Derfor er det en god ide at indlægge et kort delay inden man begynder initialiseringen af displayet og ifølge databladet er det mindst 30 millisekunder. Ellers vil displayet ikke altid modtage de første instruktioner, hvilket vil resultere i en forkert initialisering.

Cursoren:

Efter grund indstillingerne er udført kommer man til cursoren, der er en meget vigtig del af displayet. Cursoren bestemmer hvor på displayet der skal skrives, og om det skal være fra venstre mod højre eller omvendt.
Der er muligt at angive en specifik location for cursoren f.eks. 2 linie, tredje plads vil have den binære værdi: b'10010110' Som man kan se i skemaet på forrige side angiver det første ettal fra venstre, at det er display-addressen man nu sender. I displayets data blad er der på side 9 en hel oversigt over alle display adresserne.
Nu hvor man kan angive en specifik adresse for hver plads på displayet, burde man jo angive en adresse hver gang man skal skrive et bogstav, men nej. Displayet er bygget med en ekstrem smart funktion, nemlig "auto-incrementing". Denne funktion gør at hver gang man sender en karakter eller et bogstav, så finder displayet selv ud af at rykke cursoren en plads til højre. Men man skal lige huske at hvis man overskrider f.eks. første linies karakterantal (20), så fortsætter cursoren altså på den visuelle linie 3. Ligesom beskrevet i afsnittet "*2 eller 4 linier?"*.
Man kan også vælge, ikke at lave linieskift på displayet dvs. at hvis man sender 25 bogstaver til linie et, vil der kun blive vist de første 20. Men her kommer funktionen "Display shifting" ind i billedet. Ved at sende et display shift, vil displayet scrolle en plads til højre og derved vise bogstav 21 og skjule det første på linien. Ved at sende denne kommando adskillige gange vil det virke som en løbetekst, der skal selvfølgelig indsættes en eller anden form for delay, ellers vil man ikke kunne opfatte det med det blotte øje.

Busytime:

Dette er den tid det tager for displayet at udføre instruktioner ligesom ved opstarten, hvor vi indsatte et delay. På side 10 i databladet er der en oversigt over de tider det tager at udføre hver kommando og man bør opbygge sin kontroller efter disse tider for at opnå bedst mulig stabilitet.
Dog har vi ikke taget hensyn til disse pga. vi ikke belaster displayet max og at det kører upåklageligt uden delay efter hver instruktion. Desuden kræver vores programstruktur en masse assemblerkode og derfor vil vi ikke have mere end højest nødvendig unødig kode. Herunder er et udsnit af databladet som vi nogle af displayets instruktiotider: Resten kan ses i bilag 5.

Instruction	Instruction Code										Description	Execution Time(fosc= 270kHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write"20H" to DDRAM set DDRAM address to "00H" from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed	1.53ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display	39 μ s
Display											Set display(D)	

Resumé:

Den før skrevne teori omhandlende displayets opsætning, burde give en vejledende forståelse for displayets virkemåde, men der er naturligvis en masse detaljer som vi ikke vil tage med her. Det er blandt andet fordi detaljerne hænger sammen med den assemblerkode vi har skrevet til PIC'en og det er svært at give en grundlæggende forklaring på hvordan det virker, når man ikke umiddelbart kan sammenligne med selve koden. Derfor omhandler det næste kapitel den præcise opsætning af displayet via PIC'en

Kapitel

6

Programmering med PIC16F84A

Indledning:

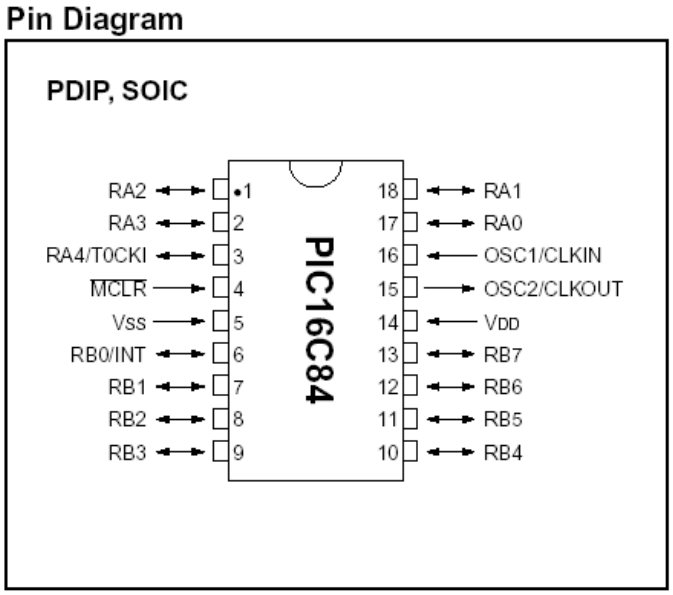
Som udgangspunkt har vi valgt at anvende denne PIC-processor frem for de mange andre varianter. Og det er af den simple grund, at det er den vi har haft undervisning omkring og som vi har tilgang til.
Desværre løb vi på et tidspunkt ind i problemer med 16F84's hukommelses begrænsning, den har nemlig kun plads til 1024 ord, hvilket vi hurtigt overskred. Men vha. lidt besparelser i assemblerkoden fik vi reduceret programmet så meget, at der kun lige var plads til vores instruktioner. Derfor var vi ved at undersøge om det var muligt at få en anden processor med mere hukommelse og flere ind/ud-gange.
Det er nemlig ikke kun hukommelse der er mangel på, når alle data ind/ud-gangene + RS/Read-Write/Enable er tilsluttet, er der kun 2 indgange tilbage til vores styring. Det vender vi tilbage til senere.
Den anden PIC vi fik fat i, vha. vejleder, er en 16C57C, med dobbelt så meget hukommelse og 8 ekstra ben, men her var instruktionssættet lidt reduceret og den brugte kun 12 instruktionsbit hvor 16F84A bruger 14 bit.
Det gav lidt problemer med kompilering af sourcen bla. Fordi den ikke kendte kommandoerne, så pga. lidt tidspres valgte vi at fortsætte fra vores udgangspunkt, nemlig PIC16F84A. Derfor har vi også kun 4 forskellige tekststrenge at vælge i mellem, til visning på displayet, fordi de to tilgængelige indgange kun giver 4 forskellige indgangs-kombinationer.

Kort om PIC'en:

En stor fordel er at den har et reduceret instruktionssæt, der er nemlig kun 35 instruktioner eller kommandoer at vælge i mellem, det giver selvfølgelig også en vis begrænsning, men det er fint til vores brug. Derudover er der som sagt en programhukommelse på 1024 ord og det passer også lige til vores krav. Det kunne bare være lidt mere spændende hvis man kunne udnytte displayet lidt mere, end bare 4 strenge tekst.

Driver til PIC-processor og display:

For at kunne opbygge et print til PIC'en bør man vel kende benforbindelserne, så de kommer herunder:

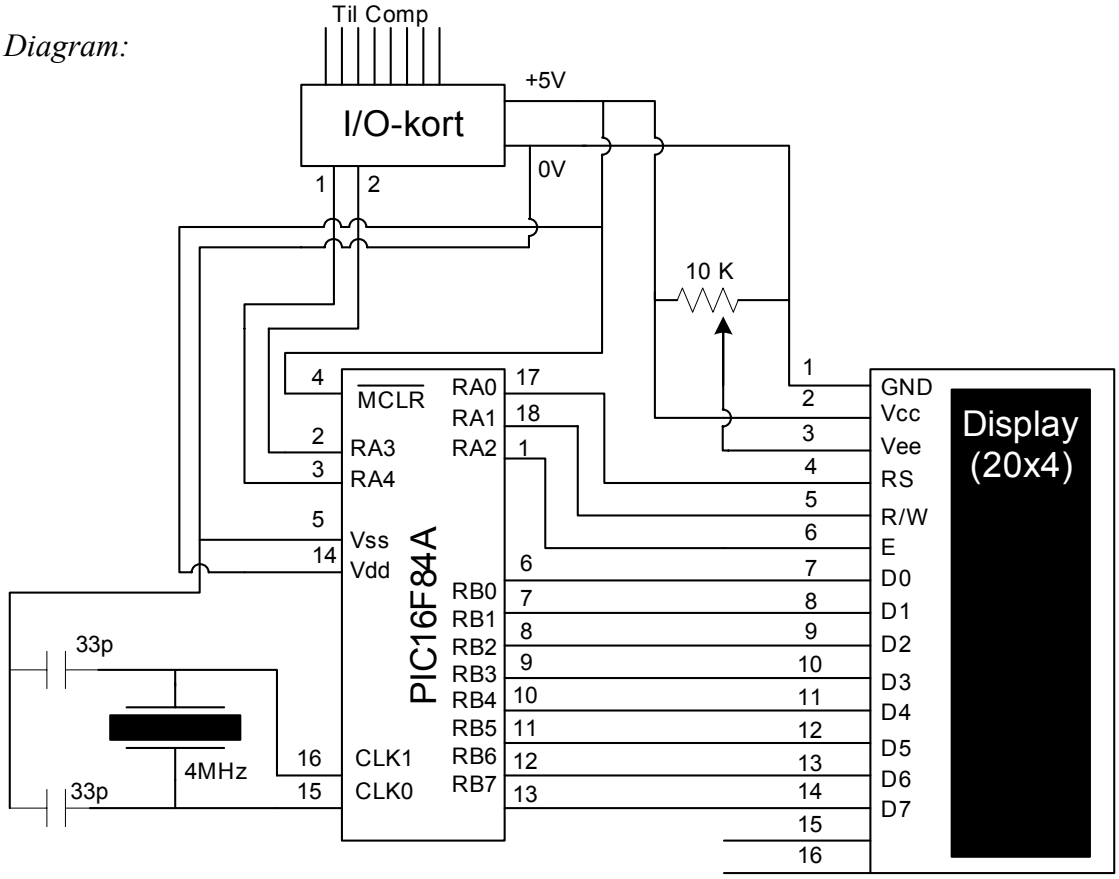


Som man kan se, er der ikke mange ben ud over ind- og udgangene. Ben 5 og 14 til forsyning og 15/16 til klok-frekvensen. Derudover er der ben 4, som er MCLR altså Memory Clear, dette ben skal altid være høj for at PIC'en er aktiv, men kan bruges som reset.

Det er meget begrænset hvilke komponenter der er nødvendig for at få PIC'en op at køre og det er jo bare godt for os som brugere.

Hele diagrammet kommer på næste side og inkluderer både PIC'en og LCD'en. Forsyningen er +5V og kan tages direkte fra det I/O kort vi bruger til styring af MATRIX-tastatur, dioder og naturligvis de 2 indgange til PIC'en.

Diagram:



Beskrivelse:

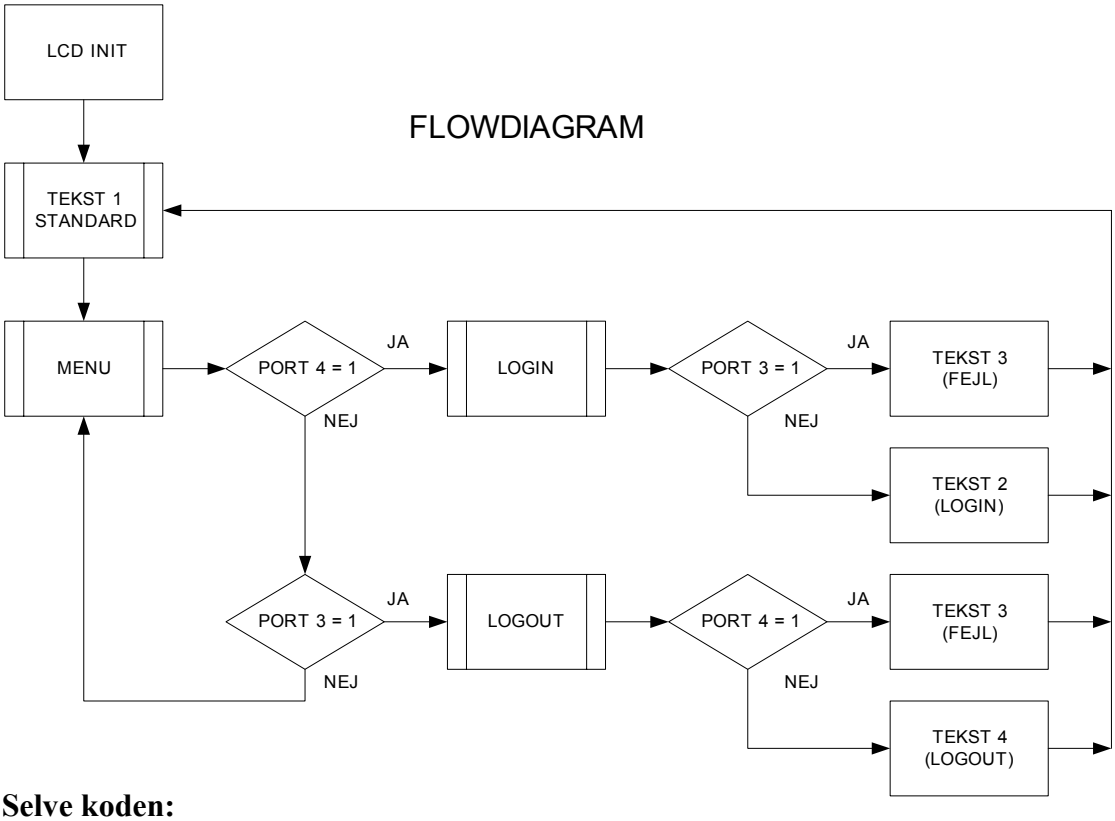
Til forsyningen på displayet er der tilsluttet et 10K Ω potentiometer til at justere kontrasten. Jo mere negativ spænding, jo skarpere kontrast. Resten af benene er digitale busser til styring og opsætning.

For overhovedet at få gang i PIC'en skal der jo forsyning til på ben 5 og 14 som vist, men der skal også en krystal til at generere en klokfrekvens som i dette tilfælde er 4 MHz. Man kan også sagtens køre ved lavere frekvenser, men så skal man anvende lidt større kondensatorer for at få et helt rent signal. Dvs. at kondensatorernes funktion er stabilisering af klokfrekvensen.

Assemblerkode til PIC16F84A

Assemblerkoden er den source der gør det muligt for brugeren/programmøren at programmere PIC-processoreren. Efter en succesfuld kompilering brændes koden ned i PIC'en via en brænder og så er den klar til brug! Assemblerkoden er vedlagt som bilag 9.
NB: Vi fandt en assembler kode på Internettet til at teste et display og vi har bygget vores kode op på grundlag af denne.

Herunder er et flowdiagram over programmets grundlæggende funktioner og der vil blive refereret til dette ved gennemgangen af kildekoden.



Selve koden:

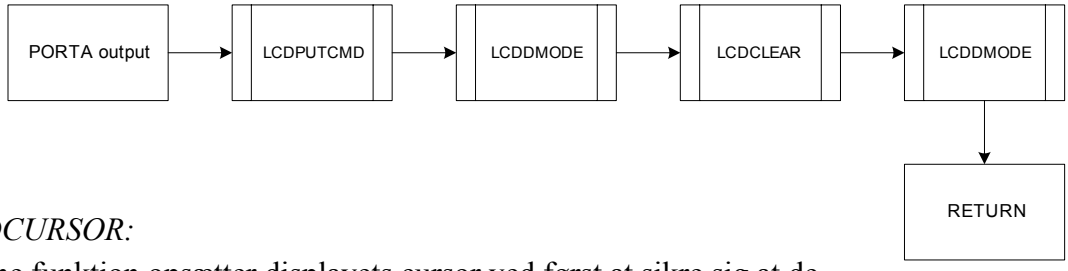
Det første man støder på er inkluderingen af p16F84A.inc der er den includefil som har de specifikke indstillinger for netop denne processor.
Dernæst skal diverse variabler defineres og kan tildeles en specifik adresse i hukommelsen, men man kan også undlade og lade processoren styre det.

Her er der bla. en variabel som man ikke umiddelbart støder på senere i programmet, men som tilhører en funktion i en senere includefil. Der er her tale om ”WCYCLE” der tilhører WAIT-funktionen der er en integreret del af kompilerens filer. Dog skal filen WAIT.inc ligge i samme mappe som den .asm fil man genererer når man gemmer sit program.

For at sikre sig at PIC’en ikke pludselig løber løbsk, kommer der efter initialiseringen af variablerne en interrupt, der henviser til start som altså er starten af vores kode.
For så at gøre vores kode mere overskuelig inkluderer vi nu nogle filer ”Bank.inc” og ”wait.inc” der hver har en vital del af programmets helhed. ”Bank.inc” indeholder nogle instruktioner til initialisering af banker altså bank0 og bank1. Disse bruges til at ”tristate” de forskellige porte, altså skift mellem input eller output.
”Wait.inc” har nogle funktioner der kan skabe nogle forsinkelser hvis der skulle være brug for det.
Hernæst skal processorens registre defineres og ”cleares”. Først skifter vi til BANK1 og begynder at definere portene som vist, hvorefter vi vender tilbage til BANK0.
Efter hele initialiseringen af processoren, variabler og registre kommer main delen hvor displayet skal initialiseres og for overskuelighedens skyld er det opdelt i funktioner.
Den første funktion eller subroutine man støder på i flowdiagrammet på forrige side, er LCDINIT der som navnet antyder er initialisering af displayet.
Selvom at LCDINIT i flowdiagrammet kun udgør en enkelt ”kasse”, indeholder den faktisk en hel del vigtige funktioner.

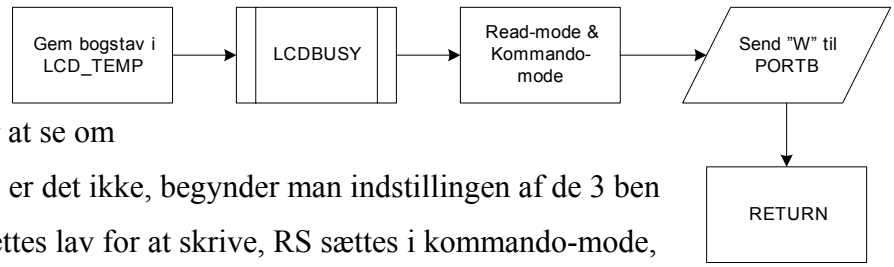
LCDINIT:
Her vil det være en god ide at kigge på *figur 1* i kapitlet om displayets opsætning, på side 33 og måske kildekoden i bilag 9 på side 118.
Det første der sker er at alle A-portene ”cleares” for at sikre sig at de står rigtigt. Dernæst venter vi et beskedent øjeblik for at displayet kan blive klar, ellers vil PIC’en bare sende instruktioner til displayet, mens det starter og det vil give en forkert initialisering. Først her starter vi med at sætte displayet op til at køre med 8 bit interface, to linier og 5 x 7 dots format, det gøres jævnfør *figur 1* ved at sende værdien 0x038 til displayet.

Så denne værdi flyttes over i arbejds-registret (w) og derefter kaldes funktionen LCDPUTCMD. Denne funktion sender instuktioner til displayet, men en beskrivelse følger først senere. (se. LCDPUTCMD).
Herefter skal displayets cursor-indstillinger nulstilles og det gøres ved at sende 0x000 til LCDCURSOR (se LCDCURSOR). Dernæst kaldes LCDCLEAR, der clearer displayet med værdien 0x001.
Eftersom vi ikke skal skrive direkte til LCD'et med et ekstern tastatur, har vi ikke brug for en cursor der viser hvor på displayet man befinder sig. Derfor nøjes vi med nu at tænde displayet med 0x004, der via LCDCURSOR tænder displayet. Herunder er et flowdiagram over LCDINIT's flow.

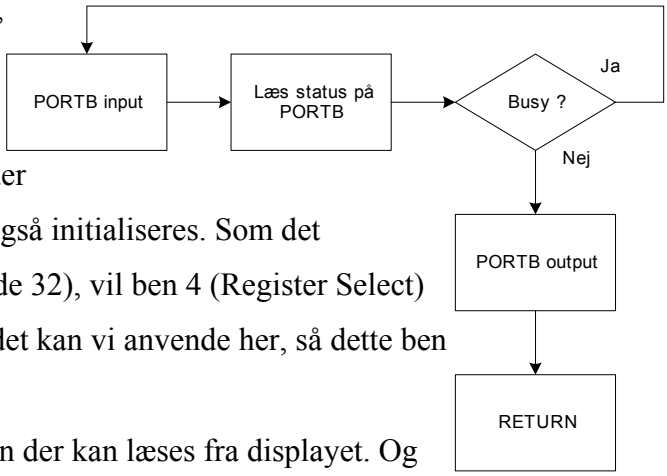


LCDCURSOR:
Denne funktion opsætter displayets cursor ved først at sikre sig at de første 3 bit ikke er sat til noget forkert. Derefter sættes bit 4 høj der ifølge *figur 1* aktiverer cursor indstillingerne. De indstillinger der sættes er overført fra "w" og herefter kaldes LCDPUTCMD.

LCDPUTCMD:
Når denne funktion kaldes, er der allerede en binær kommando i "w". Denne kommando overflyttes så midlertidig til en variabel: LCD_TEMP.
Herefter kalder man LCDBUSY (se. LCDBUSY) for at se om displayet er optaget, er det ikke, begynder man indstillingen af de 3 ben på PORTA. R/W sættes lav for at skrive, RS sættes i kommando-mode, lav, og enable sættes høj. Dernæst flytter man LCD_TEMP tilbage til "w", hvorefter "w" flyttes til PORTB og kommandoen udføres. Til sidst slåes enable fra igen.



LCDBUSY:



Normalt er alle 8 databen sat som output, men hvis man skal teste om displayet er optaget skal disse laves om til input, for at læse statusen. Så det er det første der sker her, men de 3 ben på PORTA skal også initialiseres. Som det fremgår af teoriafsnittet om displayet (side 32), vil ben 4 (Register Select) i lav tilstand aflæse displayets status og det kan vi anvende her, så dette ben sættes lav.

Næste ben, 5 Read/write, sættes høj sådan der kan læses fra displayet. Og ben 6, enable, skal naturligvis også sættes høj. Nu er vi klar til at aflæse hele PORTB, så det gør vi ved at sende PORTB's status over i "w".

Hvis denne er høj, går løkken tilbage til start af LCDBUSY og fortsætter indtil den går lav, altså nå displayet ikke er busy længere.

Når det så sker, sættes ben 5 til Write (høj) og databus PORTB sættes igen til output.

STANDARD:

Efter hele initialiseringen af displayet kommer det egentlige program, som vi styrer via de to indgange RA3 og RA4 og det er her man muligvis vil skabe sig et større overblik over det samlede program, ved at kigge på flowdiagrammet side 39.

Som man kan se, kommer man til en standard tekst der vises hver gang en bruger har logget ind eller ud og som vises så længe der ikke er aktivitet på de to indgange.

Standard-teksten (TEKST1) er "Laes kort/Tast kode!!".

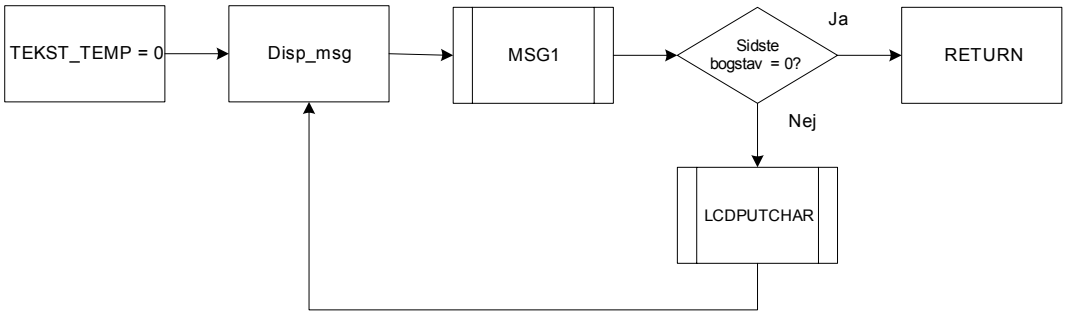
Det første der sker er et kald til LCDCLEAR, hvor efter TEKST1 kaldes. (se. TEKST1)

Når denne tekst er vist fortsætter man videre til MENU (se. MENU).

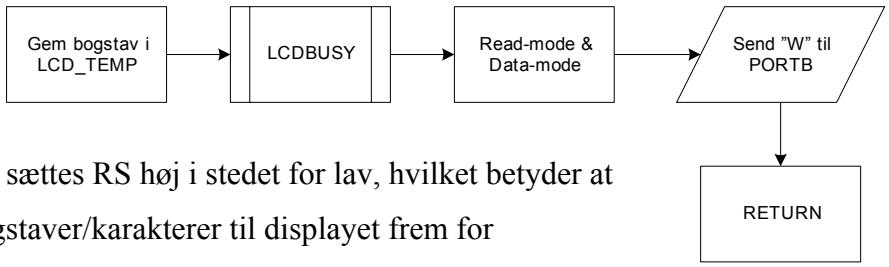
TEKST1:

0 flyttes over i TEKST_TEMP og MSG1 kaldes. MSG1 er en liste over alle de bogstaver der skal skrives på displayet. MSG1 returnerer det første bogstav "L" vi kommer tilbage til TEKST1. Her testes der om bogstavet er "0", hvis det er det er hele teksten blevet

skrevet, hvis ikke tælles kaldes LCDPUTCHAR (se. LCDPUTCHAR) der viser bogstavet på displayet. Herefter tæller TEKST_TEMP op med 1.
Og man går tilbage til TEKST1 for at fortsætte med det næste bogstav. Det sidste bogstav i MSG1 er et 0 og hvis det returneres springer man ud af løkken og returnerer til STANDARD.



LCDPUTCHAR:
Denne funktion er faktisk identisk med LCDPUTCMD, da det er samme princip de er bygget efter.



Den eneste forskel er, at ved opsætning af porte sættes RS høj i stedet for lav, hvilket betyder at man kan sende bogstaver/karakterer til displayet frem for kommandoer.

MENU:
Efter STANDARD kommer denne lille menu, der tester de to indgange for eventuelle signaler, hvis der ikke kommer nogle indgangssignaler vil menuen stå i en evig løkke og måle på indgangene. Men hvis der så kommer et signal på PORTA, 4, kaldes LOGIN (se.LOGIN). Ved signal på PORTA,3, kaldes LOGOUT. (se. LOGOUT).

LOGIN:
Først cleares displayet ved at kalde LCDCLEAR, dernæst tjekkes der om der er signal på PORTA, 3. Dvs. hvis begge indgange er høje skal funktionen FEJL kaldes. (se. FEJL).

Hvis PORTA, 3, er lav fortsætter vi med at vise TEKST2, der virker ligesom TEKST1+3+4. TEKST2 bliver vist så længe PORTA,4, er høj og fjernes først når vi via porten sætter den lav, derved returnere programmet også til STANDARD.

LOGOUT:
Denne funktion virker faktisk som ovenstående, bortset fra at vi her tester på PORTA,4, for at se om programmet skal kalde FEJL.

FEJL:
Teksten ”Forkert kort/kode” vises kun hvis begge indgange er høje. Ved at sætte en port lav returneres der til standard.

C-kode til PIC16F84A via I/O-kort

Eftersom vi kun anvender 2 udgange på I/O-kortet, bliver der ikke så meget kode i dette afsnit. Der er jo som sagt kun 4 kombinationer at vælge i mellem.
Vi vil her henvise til bilag 3 med kildekoden og funktionen ”Display” på side 91.
Funktionen er lavet sådan man at ved at kalde den, kan overføre en værdi (valg) der så henviser til de forskellige udgangskombinationer.
Helt i starten af kildekoden på side 1, bilag 3, har vi defineret de forskellige porte på I/O-kortet.

Det første der defineres er BASE adressen som er kortets adresse. Denne må selvfølgelig ikke konflikte med andre af computerens adresser, så derfor vælges der her en ledig adresse. En forklaring af I/O-kortets opsætning kommer i et andet afsnit.
Tilbage til funktionen ”Display()”, hvor der startes med at sende CWD til I/O-kortet. Altså PORTA sættes til input og PORTB+C til output. Herefter kommer de forskellige udgangskombinationer.
Hvis nu, for eksempel, tallet 2 overføres her til denne funktion vil nr. 2 if-sætning blive sand og den binære værdi b’1000000’ sendes til kortet, hvilket vil få PIC’en til at vise teksten ”Du er nu logget ind!”.

Det er linien ”`outportb(BASE + PC ,0x80 + login);`” som kalder I/O-kortet. Til sidst, alt afhængig af login’s status (1 eller 0) vil Port C, 1 også aktiveres. ”Login” er en test af om der er nogen logget ind, men det beskrives i et andet afsnit[11].
Samtidig vil funktionen ”dioder” blive kaldt med værdien 1, hvilket får den grønne diode til at lyse. Herefter kommer der et lille delay for at holde teksten og dioden aktiv længe nok til at brugeren kan nå at se hvad der står.
Til sidst bliver dioden slukket ved at sende 0x00 ud på PORTB hvor de er tilsluttet.
De andre udgangskombinationer virker på samme måde, via en overført værdi fra 1-4.

Delkonklusion:

PIC’en virker som den skal, men desværre er der kun hukommelse til at den kan indeholde fire tekst-strenge. De fire strenge er som vist herunder:

Binær:		Tekststreng:
Port 4	Port 3	
0	0	1: LAES KORT/TAST KODE!
1	0	2: DU ER NU LOGGET IND!
1	1	3: FORKERT KORT/KODE!!
0	1	4: DU ER NU LOGGET UD!

Port 3 og 4 er de to indgange på PIC’en.
Derudover er der 8 databusser og 3 funktionsben reserveret. Det giver, som vist, kun 4 binære værdier, så der er ikke mulighed for at udvide denne PIC16F84A, men der findes mange andre varianter som vi ikke har kigget på.
Dog forsøgte vi med PIC16C57C, som umiddelbart havde samme egenskaber, men ved nærmere undersøgelse og tests finder vi ud af at der faktisk er meget stor forskel på de to og derfor forbliver vi ved den først valgte.
Angående C-koden, så kan man sige at der skal utrolig lidt konfiguration til, for at kommunikere med I/O-kortet.
Derudover er der meget lidt kommunikation med selve PIC’en, men gør bare programmet mere overskueligt.

[11] Se kapitel 8 side 52

Kapitel

7

Standard matrix tastatur

Indledning:

Da vi begyndte at lege med tastaturet, tænke vi at da den havde 9 ben, måtte en af dem være input og 8 af dem output, og den så smed en høj ud på 2 ben, eller sådan noget. Men det viste sig at det ikke at være rigtigt. Da den laver en lus mellem 2 ben, når man trykker på en tast, et såkaldt matrixtastatur.
Det andet ville dog være det nemmeste, så vi bestilte et andet tastatur, men det viste sig også bare at dette var med matrix tilslutning. Så det endte med at vi nu havde to matrixtastaturer.
Siden man nu sælger så mange tastaturer med matrix, så måtte der jo også være en eller anden smart måde at anvende det på. I første omgang havde vi tænkt på PEEL-kredsen, som den bedste løsning, men vi havde fuldstændig overset at vi kunne bruge vores I/O-kort til styringen. Men da vi nu også skulle bruge I/O kortet til vores display, kom vi i tanke om, at der var en ledig 8 bits port der passende kunne bruges til vores fordel.

KEY NO.	OUTPUT							
	C_1	C_2	C_3	R_1	R_2	R_3	R_4	SC
1	●			●				
2		●		●				
3			●	●				
4	●				●			
5		●			●			
6			●		●			
7	●					●		
8		●				●		
9			●			●		
*	●						●	
0		●					●	
#			●				●	
PIN NO.	1	2	3	7	6	5	4	8

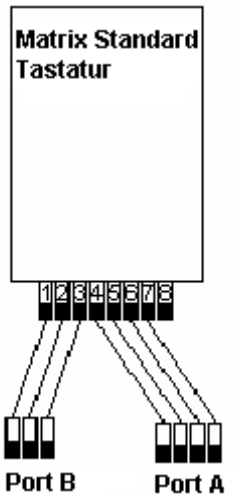
Vi benytter tastaturet til at brugeren kan skrive personens kode nå han kommer. Tastaturet har matrix tilslutning og det betyder at nå man eks. Trykker 1, så laver den forbindelse mellem ben 1 og ben 7. Se diagrammet her til højre:

Denne måde at sætte tastaturet til gav en del problemer i starten, men vi fik det løst ved at vi først sender en høj ud på ben 1, og så måler på benene 7,6,5 og 4, herefter sender vi en høj ud på ben 2, og måler igen på ben 7,6,5 og 4 og til sidst sender vi en høj ud på ben 3, og derefter igen måler på 7,6,5 og 4.

Da vi gør dette helt vild hurtigt kan programmeret nå at opfange hvis brugeren taster på en knap.

Efter man har trykket på en knap har vi så sat en lille forsinkelse ind, for ellers skriver man det, man har trykket på rigtigt mange gange.

Vi har sat tastaturet til på denne måde, som vist her til højre:

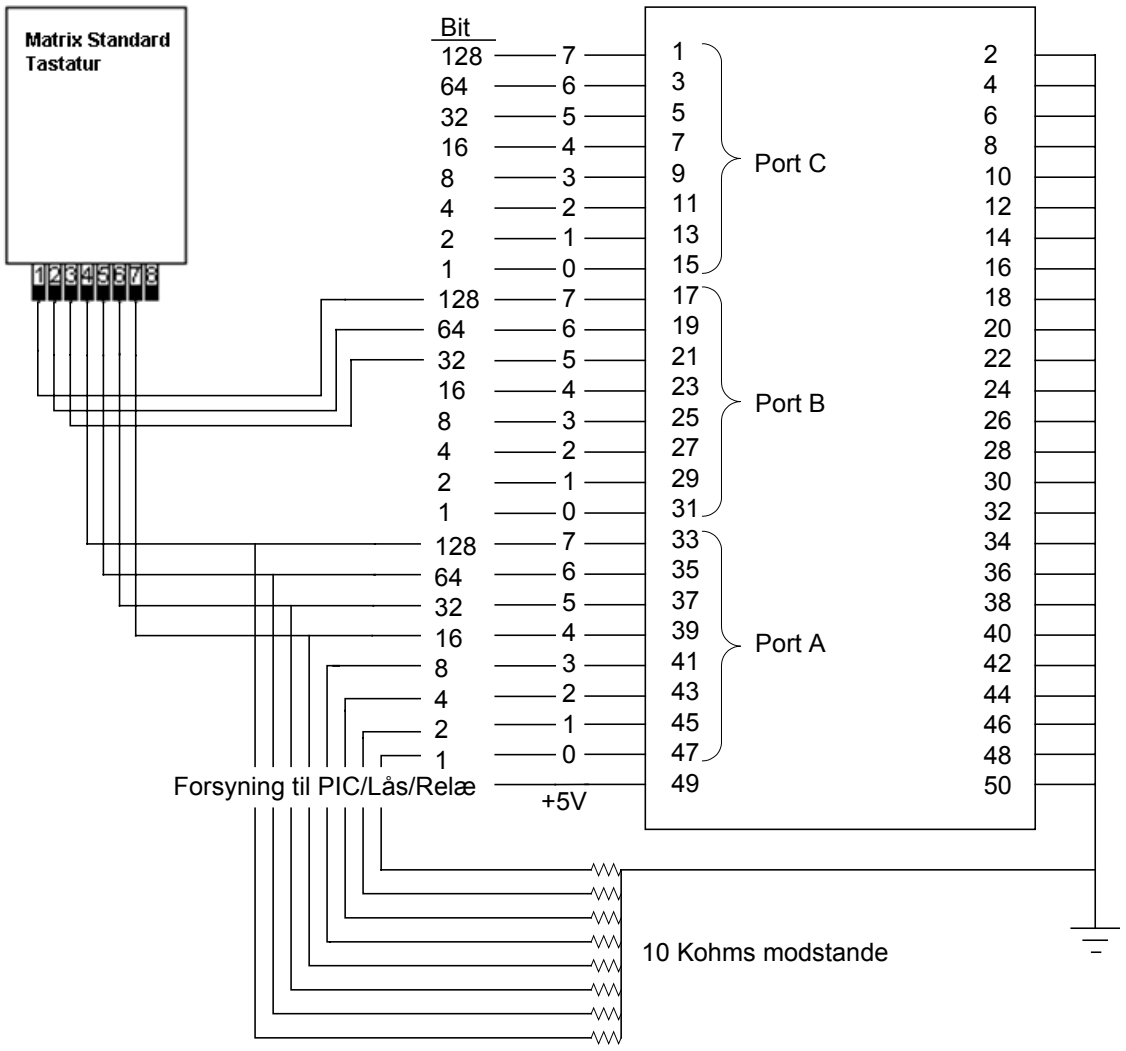


Neden under er en oversigt og matrix'et og bit-kombinationen på de forskellige porte.

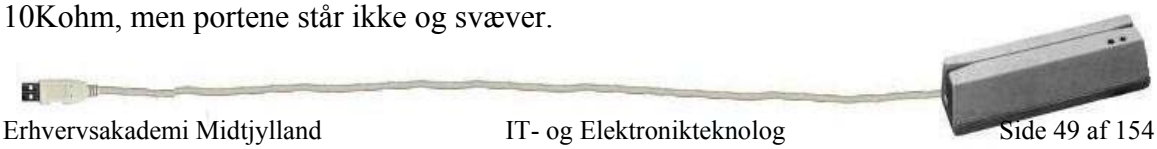
Port B / Port A	128	64	32
128	*	0	#
64	7	8	9
32	4	5	6
16	1	2	3

Kommunikation:

Kommunikationen mellem tastatur og C, forgår via vores I/O kort som vist her under. Hvor vi har sat tastaturet til på 2 porte, (A,B) således at vi smide noget ud på den ene port, og læser så på den anden. Da tastaturet laver en forbindelse mellem 2 ben, en slags lus, har vi lavet således det smider 128 ud på port B, derefter læser vi på port A. For at finde ud af om der kommer noget retur.
Ved at gøre dette finder vi ud af hvilket binær tal som kommer retur på port A.



På indgangene har vi sat 10 Kohms modstande for at trække dem ned, da de ellers vil stå og svæve. Da indgangs impedansen er på 262Kohm, betyder dette ikke en hel masse med 10Kohm, men portene står ikke og svæver.



Programmering:

Vi sender 128 ud til port b (80 i hex er lig med 128 i 10 tals systemet).

```
outportb(BASE + PB ,0x80);// B'10000000'
```

Herefter læser vi Port A.

```
pa_data = inportb(BASE + PA); // Læser fra PORTA
```

Her efter skal vi tjekke om værdien der kommer ind er 16, 23, 64 eller 128.

```
if(pa_data == 16) {
                tal = 1;
                itoa(tal, kode, 10);
                strcat(streng, kode);
                delay(50);
 }
if(pa_data == 32) {
                tal = 4;
                itoa(tal, kode, 10);
                strcat(streng, kode);
                delay(50);
}
if(pa_data == 64) {
                tal = 7;
                itoa(tal, kode, 10);
                strcat(streng, kode);
                delay(50);
}
if(pa_data == 128) {
                tal = 1;
                itoa(tal, kode, 10);
                strcat(streng, kode);
                delay(50);
}
```

Vi gentager så dette 3 gange, hvor vi i stedet for at smide 128 ud på Port B smider vi 64 ud, og så 32 ud.

Del konklusion:

Vi havde lidt svært ved at sætte os ind i matrix tilslutningen til at starte med, hvor vi bøvlet noget rundt i hvordan vi skulle lave tilslutningen, hvor vi blandt andet arbejde på noget med noget peel kredse, pic kreds og andet.
Men da vi alligevel skulle have I/O kort, med 24 digitale ind / udgange, kunne vi lige så godt bruge det til at styre tastaturet.
Da vi så med denne kunne finde ud af hvilket porte som var trukket eller hvilket som var slukket.

Kapitel

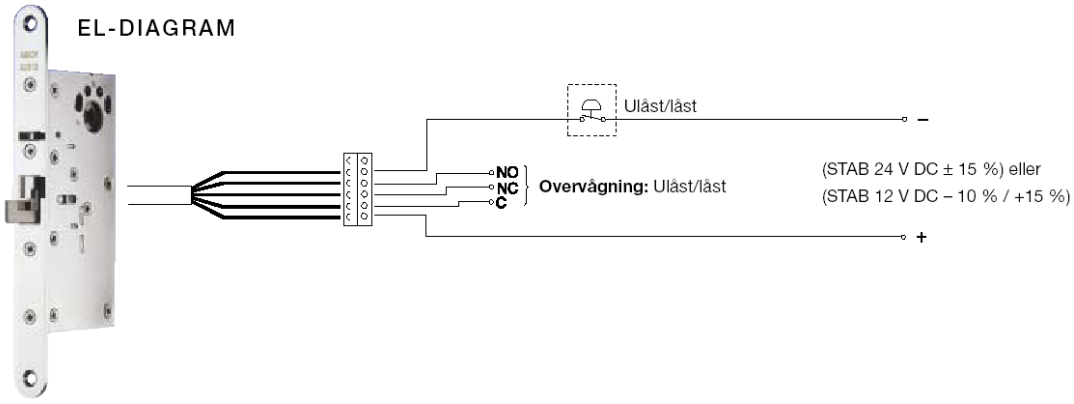
8

Dørlås (slutblik)

Da vi ikke har en slutblik, har vi lavet et relæ, hvor vi så kan trække styrestrømen til en el-slutblik. Vi har så sat vores en lys diode ind her, i stedet for, at en lås, som låser døren op og låser døren. I relæet er der mulighed for både at bruge NO & NC (Normally Open og Normally Close).

Teknisk specifikation:

Den lås vi har valgt at tage fat i, kan man få i 2 udgaver, en hvor den låser døren nå den for strøm og en hvor den er åben nå den ikke for strøm. Dette kan være til hjælp ved evt. brand hvor man skal sikre sig at folk kan komme ud, eller ved adgangskontrol, hvor det ikke er smart at en tyv kan slukke strømmen og så komme ind.
Ligeledes kan man overvåge om døren er låst op eller låst. Praktisk til eksempelvis alarm og sådan noget. Således at man ikke kan tilslutte alarmen fordi døren ikke er låst. For mere information omkring låsen se Bilag 11.

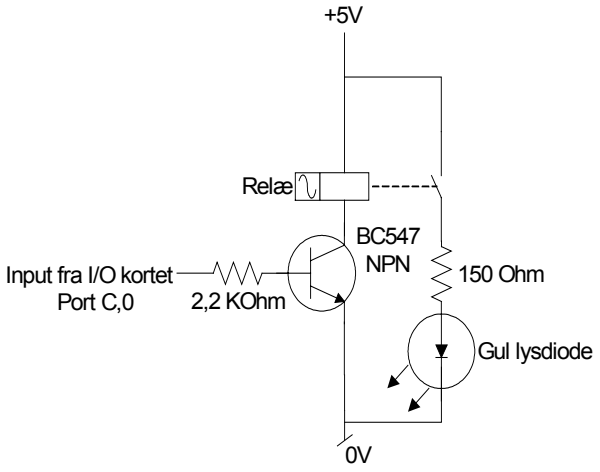


Kommunikation:

Da låsen sidder på samme port som displayet, bliver låsen styret af en global variabel, og den kan så kun skiftet hver gang displayet skifter. Dette betyder dog ikke noget, da displayet altid vil skifte, ved log ind / log ud.

Det er så lavet således at en funktion tjekker for om der er nogen som er logget ind eller nogen som er logget ud og denne funktion skriver så enten 0 eller 1 i den global variable med navnet login. Dette gør den lige før den skriver standard teksten ud på displayet, i funktionen start.

Diagram:



Relæet tager 50mA for at køre, og minimum H_{fe} på en BC547 er 110 gange forstærkning. Hvis man så tager de 50mA og dividere med Hfe, så får man hvilken strøm der skal til, for at transistoren vil være fuld åben.

$$\frac{50}{110} = 0{,}45 \approx 0{,}5mA$$

For så at være helt sikker på at transistoren er fuld åben, så har vi valgt at gange 0,5 med faktor 3. Og det giver 1,5mA. (Altså en sikkerhedsfaktor på 3)
Spændingen på en udgang på I/O kortet er 3,6 Volt, Spændingen i Base / emitter er 0.6 volt.

Der skal 1,5mA til at åbne transistoren. Dette bevirker at hvis man tager 3,6 volt og trækker de 0.6 volt over transistoren, fra(Spændingensfaldet på base/emitter), så har man spændingen over formodstanden på transistorens base.
Da der skal 1,5mA til at åbne transistoren, er det også denne strøm som der vil komme til at gå igennem formodstanden. Så kan man dividere volt, med ampere og man for så ud hvor stod modstanden skal være.

$$\frac{3{,}6-0.6}{1{,}5mA} = 2000\Omega \approx \underline{\underline{2.2K\Omega}}$$

Diodens formodstand:

Forsyningen er 5V og spændingen over dioden[12], er 2V dvs. spændingen over modstanden er 3V. Og med ohms lov får man følgende udtryk.

$$\frac{3V}{20mA} = 150\Omega$$

For at være sikker på at modstanden kan holde til den effekt vi påfører den, regner vi lige effekten ud for modstanden. Dette gør man med I * U = P, da der vil ligge 2V over lysdioden, vil resten af spændingen ligger over modstanden, dette giver så 3V, da forsyningen er på 5V. Og diodens minimum strøm er på 20mA[13] for en gul lysdiode.

$$3 * 20 = 60mW$$

Da de modstande vi har brugt kan holde til 250mW, betyder dette ikke noget.

Resumé:

Eftersom vi ikke kunne få fat i et slutblik, måtte vi simulere kredsløbet og har i stedet indsat en diode til at symbolisere låsen. Derudover har vi forsynet dioden direkte fra I/O-kortet, hvilket i praksis ikke vil være realistisk fordi en sådan lås vil kræve en større spænding og derved en ekstern forsyning.

[12] Se bilag 15 side 139
[13] Se bilag 15 side 139

Kapitel

9

Web delen

Opsætning *af Apache & PHP4:*

Vi benytter apache og php4, da begge dele er af type open source, og ligger gratis til downloading på internettet.
På henhold www.apache.org & www.php.net

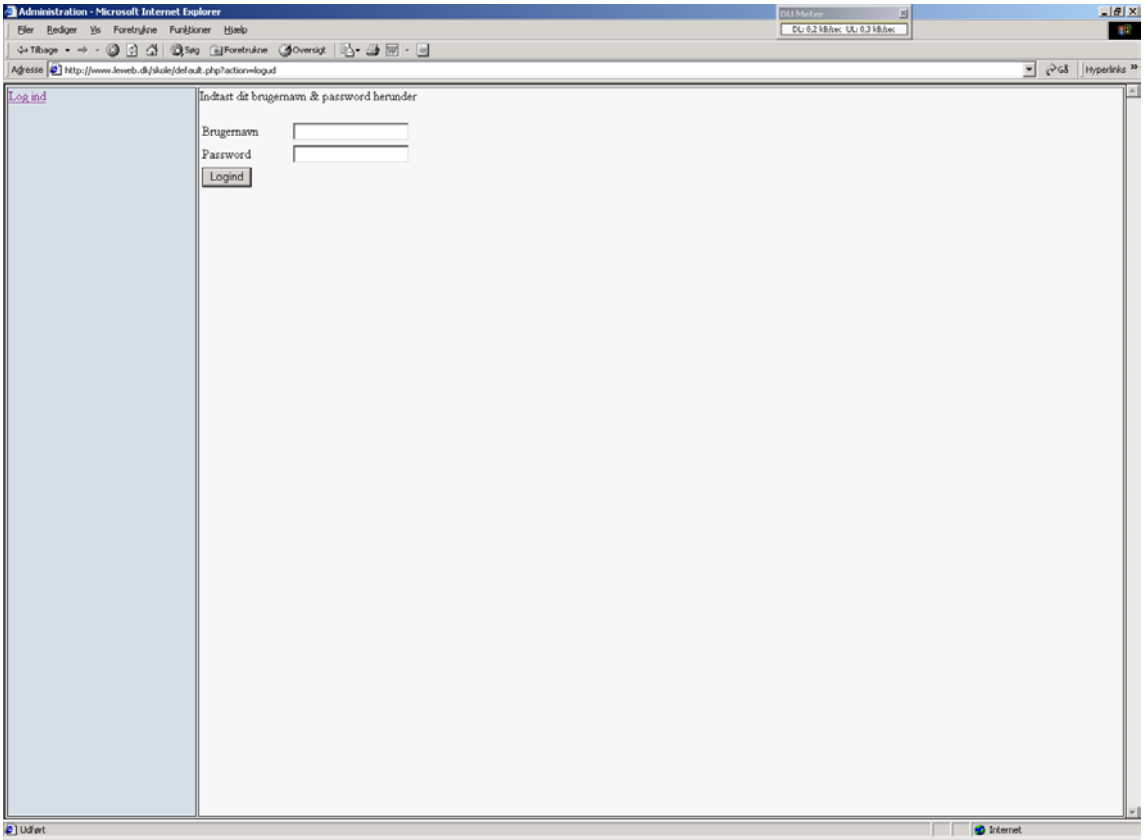
1. Dobbeltklik på filen apache_1.3.26-win32-x86-no_src.exe.
2. Tryk på knapperne "Next", vælg "I acccept..." og "Next" indtil man kommer til vinduet hvor man bliver bedt om at udfylde oplysninger om ens server ("Server Information"):
3. Her gør man følgende:
   - "Network Domain" udfylder man blot med: "localhost".
   - "Server Name" med: "127.0.0.1".
   - "Administrator's Email Address" med en emailadresse.
4. Klik "Next". Og vælg "Complete" installation.
5. Klik "Next" igen, og accepter det foreslåede bibliotek, eller vælg "Browse" for selv at angive biblioteket.
   (Bemærk at vi i resten af denne kapitel, for nemheds skyld, antager at man gemmer det i biblioteket "c:\apache").
6. Klik "Next" og derefter "Install".

Apache bliver nu installeret og starter automatisk op bagefter, med mindre at der opstår problemer under installationen.
Vi gå nu videre til php installationen:

Først skal man have udpakket php4, kopiere det over så det ligger i c:\php. Kopiér så filen php4ts.dll over i dit windows system biliotek. Der er også en ini fil med php, den skal du kopiere over i dit windows biliotek.
LoadModule php4_module c:/php/sapi/php4apache.dll
AddType application/x-httpd-php .php

Start så apache op (gøres via start – programs – apache.)
Og så skulle det gerne virke.

Vi har lavet en web-del af vores program, her kan man se loggen og hvem som er logget ind.
Vi har som førnævnt lavet løsningen i php og delt koden op i 2 filer, som er:
Default.php
Function.php
Når man går ind på webserveren får man dette billede frem, her bliver man bedt om at



taste brugernavn og password ind, dette er i vores eks, admin & test. (Brugernavn & kodeord, kan ændres i kildekoden[14].

Hjemmesiden er her delt op i 2 tabeller, den ene er til menuen, som vi kalder med menu(); den anden er til den funktion man nu vælger at kalde fra filen funktion.php.

Nå man har udfyldt brugernavn & password og trykker, skifter menuen til dette:

Hvem er logget ind
Vis loggen
Log ud

Her kan man så vælge mellem at se log filen & vis loggen, eller logge ud, hvis man forlader maskinen.

Hvis man trykker på "Hvem er logget ind" så viser den log filen, her et udkast af den:

Følgende bruger er logget ind:
Jens
Dalbjerg

Hvis man vælger Vis loggen viser den:

Log filen:
<--Log start-->
11:30:26 Start Program
21:27:07 Start Program
21:28:12 Program exit
21:28:22 Start Program
21:29:39 Created User Ole
21:30:51 Deleted User Ole
21:31:07 Program exit
21:35:51 Start Program
21:36:15 Created User Bent
21:36:17 Program exit
8:20:01 Start Program
8:20:32 Created User Jens
8:21:01 Program exit
8:22:11 Start Program
8:22:29 Created User Ole

[14] Se Bilag 7 side 106.

Gennemgang af kildekoden

I hoved filen starter med session_start(); dette er noget man skal huske at includere i alle de filer hvor man skal bruge sessions. Session bruges til at gemme en værdi på webserveren, således at man kan hente denne værdi frem en anden gang. Dette er for eksempel smart i en log ind frekvens hvor den skal huske på om en bruger er logget ind eller ikke.

For at inkludere filer, skal bruger man kommandoen include(); dette er praktisk der man så kan ligge funktioner i en fil for sig selv, således at hvis man har mange filer, men gerne vil bruge de samme funktioner i mange filer, så kan man ligge lave en fil, med funktioner i.

Der er 3 funktioner i denne php kode, de ligger alle i filen function.php.

Vislog()
Den åbner tekst filen log.txt og skriver hver enkel linie ud en for en, til den når bunden. Vi gør så brug af en intern funktion i php ved navn htmlentities(), denne gør sådan at teksten bliver html venlig, eks hvis nu der i en linie havde stået <> så vil den bytte ud med html koden for < som er < således at den viser koden, ellers vil browseren tro det er noget html, og derved ikke vise koden.

Loggedinuser()
Denne funktion åbner filen brug.txt, som er brugerfilen, denne fil henter den ind i et array, linie for linie, således at den finder ud af kun at skrive de bruger ud som i status har 1 i værdi, dem med 0 springer den over, og går så videre til den næste. Således for man skrevet alle bruger ud som er logget ind i systemet.

Menu()

Funktionen til at udskrive menuen med. Den tjekker for om en bruger er logget ind, og er man det skriver den de punkter ud som hører til dette, som er : Hvem er logget ind, Vis loggen & Log ud. Ellers skriver den menu punktet ”log ind” ud.

Derud over har vi lavet noget java script, til login.

```
<SCRIPT LANGUAGE="JavaScript">
<!--
 function formCheck ()
 {
     if (document.Logind.brugernavn.value == "")
     {
     alert("Skriv venligst et brugernavn");
          document.Logind.brugernavn.focus();
         return false;
     }
     if (document.Logind.brugernavn.value == "Indtast dit brugernavn her")
     {
     alert("Skriv venligst et brugernavn");
          document.Logind.brugernavn.focus();
         return false;
     }
     if (document.Logind.password.value == "")
     {
     alert("Skriv venligst et password");
          document.Logind.password.focus();
         return false;
     }
}
// -->
</SCRIPT>
```

Hvor den går ind og tjekker for om man har tastet noget i brugernavn & password, i formularen.
Har man ikke, vil den poppe op med en alert:

Her ser du den, hvis man ikke har skrevet et brugernavn.

Resumé

Vi har udarbejdet løsningen i PHP fordi vi så er fri for at C++ skal skrive i 2 filer, men bare kan nøjes med en. F.eks. vil JAVA-Scriptet gerne have sideløbende fil til variabler. Logfilen kan heller ikke umiddelbart læses via JAVA-Script når formatet er txt, så også her er det nemmere at anvende PHP.
Og da PHP, ligner C++ var det også lige til at gå til. Backend er en Apache webserver, den er både den meste brugte webserver på nettet, gratis og af typen open source.

Konklusion

Trods det, at vi havde opbygget et meget stort projektoplæg, synes vi selv, er det faktisk stort set lykkedes at løse de opstillede problemstillinger. Dog afviger det færdige produkt lidt fra det produkt vi havde forestillet os, men det skyldes bla. at vi jo ikke kendte løsningen på forhånd og derfor måtte arbejde ud fra de ideer vi havde pt. Eller dvs. produktet har de funktioner og egenskaber som først planlagt, men den egentlige opbygning har en lille afvigelse. F.eks. gik vi ud fra at kommunikationen med displayet skulle foregå via parallelporten (LPT1) og så derfra styres af en PIC-processor, men da vi efterhånden gik lidt fast i parallelkommunikation, gik vi over til I/O-kortet.
Det smarte her, var at vi ikke kunne undvære I/O-kortet alligevel. Tastatur, dioder og lås skulle alligevel styres derigennem og derfor kunne vi også lige godt bruge I/O-kortets porte til at styre displayet, i stedet for parallel-portens.
Udover problemet m. parallelporten, har vi ikke fået de to PIR sendere med, som beskrevet i projektbeskrivelsen. Selvom de står som en sekundær opgave, har vi valgt ikke at tage dem med fordi de principielt virker ligesom vores afsnit om låsen. Dog med lidt ekstra sikkerhedsbestemmelser osv.
I stedet for har vi opbygget det tertiære mål, hjemmesiden, fordi det giver et godt billede af hvordan systemet kan anvendes i praksis. Siden kan vise logfil og hvem der er logget via loginsystemet, så derfor er der dog da stadig plads til en masse udbyggelse på, men det har tidsplanen ikke tilladt.
Tidsplanen har vi fulgt slavisk og derfor har vi hele tiden haft et godt overblik over hvor langt vi var kommet i projektet. Selvom vi til tider har været bagefter, har vi kompenseret med lidt ekstra aftenarbejde for at komme tilbage på ret køl.
Selve brugerdelen, altså oprettelse og redigering af brugere i systemet, virker som det skal, men eftersom vi kun har 1 magnetkortlæser er der ikke mulighed for at eksekvere login-programmet samtidig med at der oprettes brugere. Det skyldes at begge programmer vil læse fra kortlæseren og derfor ved CPU'en ikke hvilket program den skal sende de opsamlede data til.

Dvs. programmerne kan godt køre samtidig som planlagt og man kan i brugeradministrationen se logfilen og de brugere der er oprettet, men det optimale ville være et system med to magnetkortlæsere.
Loginprogrammet skal i praksis ikke have nogen form for grafisk brugerflade, fordi det bare skal køre i baggrunden og så styrer man det hele via brugeradministrationen, men vi har for præsentationsskyld lavet nogle få linier, der viser hvor i programmet man befinder sig.
Slutblikket, der styres af loginprogrammet kunne ikke fremskaffes pga. budgettet eftersom en sådan enhed er meget dyr, så derfor har vi opstillet et relæ og en diode for at symbolisere låsen. Det er her den gule diode der er tale om, og når den lyser skal det opfattes som en åben lås og omvendt. Spænding til dioden tages direkte fra I/O- kortet, men i praksis vil et slutblik kræve en større spænding og derfor en ekstern strømforsyning.

Som beskrevet i de foregående kapitler, skal man have et magnetkort og en adgangskode for at kunne logge ind, derfor mener vi at denne løsning vil kunne øge sikkerheden og gøre det nemmere for den retmæssige bruger at skabe adgang til lokalet.

Kildeliste

Display:
http://home.iae.nl/users/pouweha/lcd/lcd2.shtml
How to use Intelligent L.C.D.s Part 1 www.maxmon.com
How to use Intelligent L.C.D.s Part 2 www.maxmon.com

PIC16C84A:
An Introduction to PIC Microcontrollers af R.A. Penfold
Datablad over PIC16C84A

C-kode:
Turbo C++ af Finn Elvekjær

Serielkommunikation:
http://eam.oursite.dk/kurser/ite21-2/filer/termpoll.c
http://eam.oursite.dk/kurser/ite21-2/filer/CP_serial.pdf

Web:
http://www.sfu.ca/acs/javascript/

Generelt:
Analog teknik 2. udgave af Industriens Forlag
Digital teknik 2. udgave af Industriens Forlag

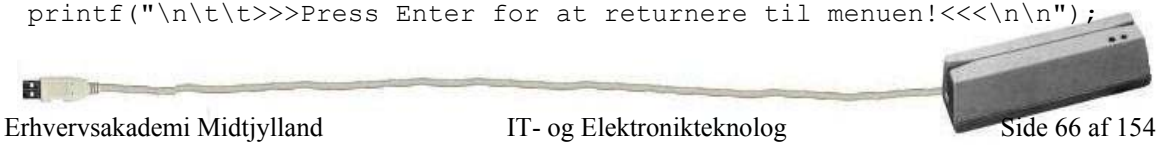
Komponentliste

Enhed:	Type:	Antal:
Til I/O-kort		
ACL7124	I/O kort	1
10 Kohm	Modstand	8
Standard Diode	Rød	1
56 Ohm	Modstand	1
Standard Diode	Grøn	1
82 Ohm	Modstand	1
Til PIC'en		
PIC16C84A	Micro Processor	1
4 Mhz	Krystal	1
33pF	Kondensator	2
10 Kohm	Potentiometer	1
Til Display		
20X4 LCD	Display	1
Til Tastatur:		
Numerisk Matrix	Tastatur	1
Til lås:		
SGR462	Relæ	1
BC547	Transistor	1
1N4007	Diode	1
2,2 Kohm	Modstand	1
Standard Diode	Gul	1
Polytop IP65	Elektronik boks	1

Bilag 1

-Kildekode til Brugeradministration

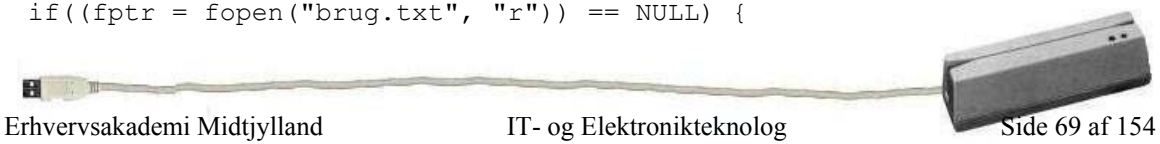
```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <dos.h>

#define PORT1 0x3F8

struct bruger_info
{
 char STbruger[40], STkortnummer[40], STkode[40], STlog[40];
 struct bruger_info *naeste;
};
struct bruger_info *foerste, *sidste, *ny;

int vislog() {
  FILE *fptr;
  char linie[81];
  int i = 0;
  if((fptr = fopen("log.txt", "r")) == NULL) {
   printf("\nFilen 'log.txt' kan ikke åbnes for læsning.");
   menu();
  }
  while(fgets(linie, 81, fptr) !=NULL) {
   if(i++ < 20)
     printf("%s", linie);
   else {
     printf("%s", linie);
     printf("\n\t\t>>>FORTSÆT: Press Enter     MENU: Press Escape<<<\n\n");
     if(getch() == 13)
       i = 0;
     else if(getch() == 27){
       fclose(fptr);
       menu();
     }
   }
 }
 printf("\n\t\t>>>Press Enter for at returnere til menuen!<<<\n\n");
```



```
if(getch() == 13)
  close(fptr);
  return 0;
}

int skriv_til_log(char *beskrivelse, char *addlog) {
  FILE *fptr;
  struct time t;
  struct date d;
  gettime(&t);
  getdate(&d);

  if((fptr = fopen("log.txt", "a")) == NULL) {
    printf("\nFilen 'log.txt' kan ikke åbnes for tilføjelser.");
    menu();
  }
  fprintf(fptr, "%2d-%02d-%02d %2d:%02d:%02d", d.da_day, d.da_mon, d.da_year, t.ti_hour, t.ti_min,
t.ti_sec);
  fprintf(fptr, " ");
  fputs(beskrivelse, fptr);
  fputs(addlog, fptr);
  fputs("\n", fptr);
  fclose(fptr);
  return 0;
}

int del_bruger(char *bruger)
{
  if(foerste==NULL)
  return 0;
  sidste=NULL;
  ny=foerste;
  while(strcmp(ny->STbruger,bruger)!=0) {
    sidste=ny;
    ny=ny->naeste;
    if(ny==NULL) return 0;
   }
   if(sidste!=NULL) {
```

```
    sidste->naeste=ny->naeste;
    free(ny);
  }
  else {
    foerste=ny->naeste;
    free(ny);
   }
   return 0;
}

int opdatefil()
{
  char VAnavn[81];
  char ch;
  int i = 0;
  int s_temp[81];
  int VAkortnummer, VAkode;
  FILE *fptr;
  struct bruger_info *ptr = foerste;

  if((fptr = fopen("brug.txt", "w")) == NULL) {
    printf("Filen brug.txt kunne ikke åbnes");
    skriv_til_log("Programfejl -> ", "Kunne ikke finde brug.txt");
    exit(1);
  }
  while(ptr != NULL) {
    if(i++ != 0)
      putc('\n', fptr);
      fputs(ptr->STbruger, fptr);
      putc(':', fptr);
      fputs(ptr->STkortnummer, fptr);
      putc(':', fptr);
      fputs(ptr->STkode,fptr);
      putc(':', fptr);
      fputs(ptr->STlog,fptr);
      putc(':', fptr);
      ptr = ptr->naeste;
    }
```

```
  fclose(fptr);
  return 0;
}

int sletbruger()
{
  char soegte_navn[81];
  int fundet = 0;
  printf("\n\n\t\tSkriv navnet på brugeren du vil slette: ");
  scanf("%s", soegte_navn);
  fundet = soegbruger(soegte_navn);
  if(fundet == 0) {
    printf("\nBrugeren findes ikke!\n\n\n");
    printf("\n\t\t>>>Press Enter for at returnere til menuen!<<<\n\n");
    if(getch() == 13)
      menu();
    else
      exit(0);
  }
  del_bruger(soegte_navn);
  printf("\nBrugeren %s er slettet!\n\n\n", soegte_navn);
  printf("\n\t\t>>>Press Enter for at returnere til menuen!<<<\n\n");
  if(getch() == 13)
    opdatefil();
  skriv_til_log("Slettede bruger: ", soegte_navn);
  return 0;
}

int hentbruger()
{
   FILE *fptr;
   char ch;
   char linie[81];
   char s_temp[81];
   int j = 0, i = 0;
   foerste = NULL;

  if((fptr = fopen("brug.txt", "r")) == NULL) {
```



```
  printf("\nFilen brug.txt kan ikke åbnes.");
  skriv_til_log("Programfejl -> ", "Kunne ikke finde brug.txt");
  exit(1);
}
while(fgets(linie, 81, fptr) != NULL) {
  ny = malloc(sizeof(struct bruger_info));
  if(foerste == NULL)
    foerste = sidste = ny;
  else {
    sidste->naeste= ny;
    sidste = ny;
  }
while((ch = linie[i++]) != ':') //Brugernavn
  s_temp[j++] = ch;

s_temp[j] = '\0';
strcpy(sidste->STbruger, s_temp);
j = 0;

while((ch = linie[i++]) != ':') //Kortnummer
  s_temp[j++] = ch;

s_temp[j] = '\0';
strcpy(sidste->STkortnummer, s_temp);
j = 0;

while((ch = linie[i++]) != ':') //Kode
  s_temp[j++] = ch;

s_temp[j] = '\0';
strcpy(sidste->STkode, s_temp);
 j = 0;

while((ch = linie[i++]) != ':') //Status
  s_temp[j++] = ch;

s_temp[j] = '\0';
strcpy(sidste->STlog, s_temp);
```

```
  j = i = 0;
  sidste->naeste = NULL;
  }
  fclose(fptr);
    return;
}

int visbruger()
{
  struct bruger_info *ptr = foerste;
  int i = 0;
  printf("Følgende brugere er oprettet i systemet:\n");
  while(ptr != NULL) {
  if(i < 9) {
    printf("%s : %s : %s : %s\n", ptr->STbruger, ptr->STkortnummer, ptr->STkode, ptr->STlog);
    ptr = ptr->naeste;
      i++;
    if(ptr == NULL)
      break;
  }
  else {
    printf("%s : %s : %s : %s\n", ptr->STbruger, ptr->STkortnummer, ptr->STkode, ptr->STlog);
    printf("\n\t\t>>>FORTSÆT: Press Enter     MENU: Press Escape<<<\n\n");
    if(getch() == 13)
      i = 0;
    else if(getch() == 27)
      menu();
    }
   }
   printf("\n\t\t>>>Press Enter for at returnere til menuen!<<<\n\n");
   if(getch() == 13)
    return 0;
}
```

```
int soegbruger(char bruger[81])
{
  struct bruger_info *ptr = foerste;

  while(ptr != NULL)
  {
    if(strcmp(ptr->STbruger, bruger) == 0)
      return 1;
    else
      ptr = ptr->naeste;
  }
  return 0;
}

int redigerebruger()
{
  char VAnavn[81], nyVAkortnummer[81], nyVAkode[81], nyVAstatus[81];
  int fundet = 0;

  printf("\n\nBrugerens navn: ");
  gets(VAnavn);
  fundet = soegbruger(VAnavn);
  if(fundet == 0) {
    printf("\nBrugeren findes ikke!\n\n");
    printf("\n\t\t>>>Press Enter for at returnere til menuen!<<<\n\n");
    if(getch() == 13)
      menu();
    else
      exit(0);
  }
  printf("Læs brugerens kort: ");
  strcpy(nyVAkortnummer, laes_kort());
  printf("Brugerens nye kode: ");
  gets(nyVAkode);
  del_bruger(VAnavn);
  opdatefil();
  opretbruger(VAnavn, nyVAkortnummer, nyVAkode, 0);
  printf("\n\nRedigering af %s fuldført\n\n", VAnavn);
```

```
  printf("\n\t\t>>>Press Enter for at returnere til menuen!<<<\n\n");
  if(getch() == 13) {
    skriv_til_log("Redigering af bruger: ", VAnavn);
    return 0;
    }
  else
    exit(0);
}

int laes_kort() {
  char s_temp[81];
  int c, j = 0;
  char ch;
  outportb(PORT1 + 1 , 0);
  outportb(PORT1 + 3 , 0x80);
  outportb(PORT1 + 0 , 0x0C);
  outportb(PORT1 + 1 , 0x00);
  outportb(PORT1 + 3 , 0x03);
  outportb(PORT1 + 2 , 0xC7);
  outportb(PORT1 + 4 , 0x0B);
  do {
    c = inportb(PORT1 + 5);
    if (c & 1) {
      ch = inportb(PORT1);
      s_temp[j++] = ch;
    }
  }
  while (ch !='?');
  s_temp[j] = '\0';
  printf("\n");
  return s_temp;
}
```

```
int opret()
{
  char VAnavn[81], VAkortnummer[81], VAkode[81];
  int fundet = 0;
  printf("\n\nBrugerens navn: ");
  gets(VAnavn);
  fundet = soegbruger(VAnavn);
  if(fundet == 1) {
    printf("\nBrugeren findes alledere!\n\n\n");
    printf("\n\t\t>>>Press Enter for at returnere til menuen!<<<\n\n");
    if(getch() == 13)
      menu();
    else
      exit(0);
   }
  printf("Læs brugerens kort: ");
  strcpy(VAkortnummer, laes_kort());
  printf("Brugerens kode: ");
    gets(VAkode);
  opretbruger(VAnavn, VAkortnummer, VAkode, 0);
  printf("\n\nBrugeren %s er oprettet!\n\n", VAnavn);
  printf("\n\t\t>>>Press Enter for at returnere til menuen!<<<\n\n");
  if(getch() == 13) {
    skriv_til_log("Oprettelse af bruger: ", VAnavn);
    return 0;
    }
  else
    exit(0);
}
```

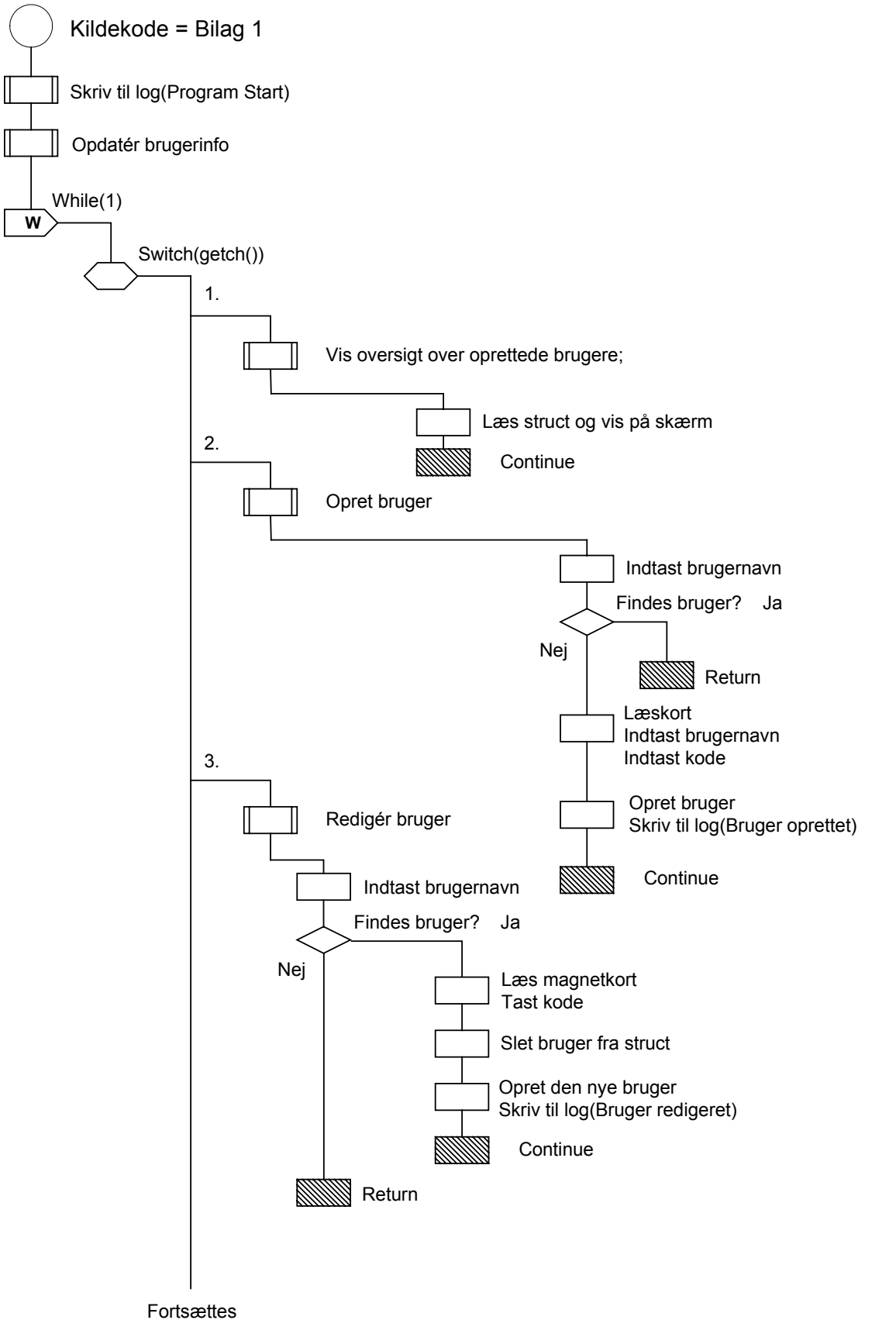
```
int opretbruger(char *VAnavn, char *VAkortnummer, char *VAkode)
{
  char ch;
  FILE *fptr;

  if((fptr = fopen("brug.txt", "a")) == NULL) {
   printf("Filen brug.txt kunne ikke åbnes");
   skriv_til_log("Programfejl -> ", "Kunne ikke finde brug.txt");
   exit(1);
  }
  putc('\n', fptr);
  fputs(VAnavn, fptr);
  putc(':', fptr);
  fputs(VAkortnummer, fptr);
  putc(':', fptr);
  fprintf(fptr, "%s", VAkode);
  putc(':', fptr);
  putc('0', fptr);
  putc(':', fptr);
  fclose(fptr);

  ny = malloc(sizeof(struct bruger_info));
  if(foerste == NULL)
   foerste = sidste = ny;
  else {
   sidste->naeste= ny;
   sidste = ny;
  }
  strcpy(sidste->STbruger, VAnavn);
  strcpy(sidste->STkortnummer, VAkortnummer);
  strcpy(sidste->STkode, VAkode);
  sidste->naeste = NULL;
  return 0;
}
```

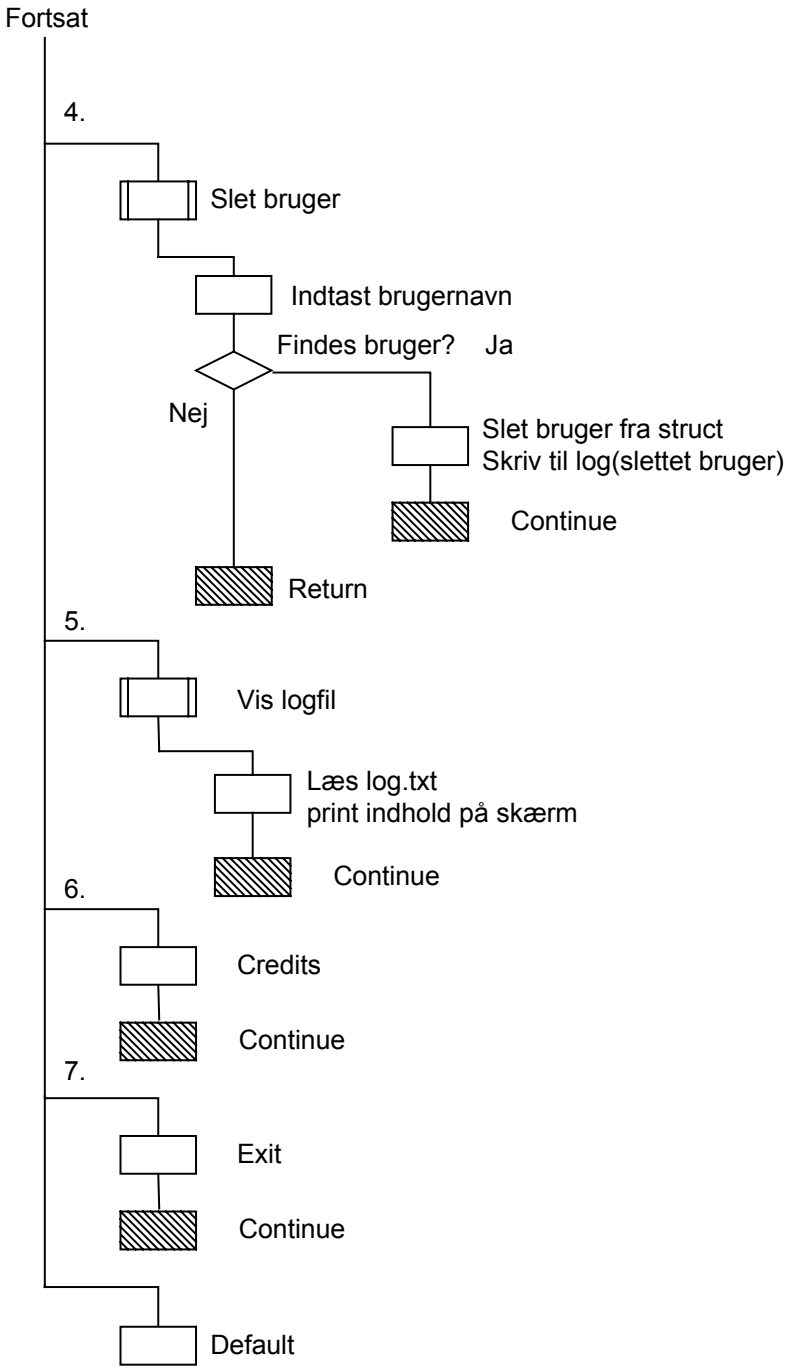
```
int credits()
{
 printf("\n\n\n\n\n\n\n\n\n\n\n\t*************************************************************\n");
 printf("\t* Copyright 2003.                                           *\n");
 printf("\t* Udarbejdet af Kenneth Dalbjerg & Ole Rosengreen ITE 2.1.  *\n");
 printf("\t* All rights reserved!                                      *\n");
 printf("\t*************************************************************\n");
 printf("\n\t>>>Press Enter for at returnere til menuen!<<<\n\n");
 if(getch() == 13)
  menu();
 else
  return;
}
int menu() {
 int i = 0;
 int valg;
 hentbruger();
 while(1) {
  clrscr();
  printf("\n\n\n\n\n\n\n\n\n\n\n\t\t\t*************************\n");
  printf("\t\t\t*  1. Vis brugere        *\n");
  printf("\t\t\t*  2. Opret bruger       *\n");
  printf("\t\t\t*  3. Redigér bruger     *\n");
  printf("\t\t\t*  4. Slet bruger        *\n");
  printf("\t\t\t*  5. Vis log            *\n");
  printf("\t\t\t*  6. Credits            *\n");
  printf("\t\t\t*  7. Exit               *\n");
  printf("\t\t\t************************* ");
  printf("\n\n\n\n\n\n\n\n\n\nVælg venligst i menuen oven for: ");
  switch(getch())
  {
   case '1':
    clrscr();
    visbruger();
    continue;
   case '2':
    clrscr();
    opret();
```

```
      continue;
      case '3':
        clrscr();
        redigerebruger();
        continue;
      case '4':
        clrscr();
        sletbruger();
        continue;
      case '5':
        clrscr();
        vislog();
        continue;
      case '6':
        clrscr();
        credits();
         continue;
      case '7':
        clrscr();
        skriv_til_log("Program ", "exit");
        exit(0);
        continue;
      default:
        printf("\nDu valgte forkert prøv venligst igen!!!");
        continue;
    }
  }
}



void main(void)
{
  skriv_til_log("Start ", "Program");
  menu();
}
```

Bilag 2

-Flowdiagram til Brugeradministration


Kildekode = Bilag 1
Skriv til log(Program Start)
Opdatér brugerinfo
While(1)
W
Switch(getch())
1.
Vis oversigt over oprettede brugere;
Læs struct og vis på skærm
Continue
2.
Opret bruger
Indtast brugernavn
Findes bruger? Ja
Nej
Return
Læskort
Indtast brugernavn
Indtast kode
Opret bruger
Skriv til log(Bruger oprettet)
Continue
3.
Redigér bruger
Indtast brugernavn
Findes bruger? Ja
Nej
Læs magnetkort
Tast kode
Slet bruger fra struct
Opret den nye bruger
Skriv til log(Bruger redigeret)
Continue
Return
Fortsættes


Fortsat
4.
Slet bruger
Indtast brugernavn
Findes bruger?
Ja
Nej
Slet bruger fra struct
Skriv til log(slettet bruger)
Continue
Return
5.
Vis logfil
Læs log.txt
print indhold på skærm
Continue
6.
Credits
Continue
7.
Exit
Continue
Default

Bilag 3

-Kildekode til Loginstyring

```
# define BASE 0x2C0 // define base address
# define PA 0x0 // Port A
# define PB 0x1 // Port B
# define PC 0x2 // Port C
# define CWD 0x90 //PORT A INPUT, PORT B/C OUTPUT
# define PORT1 0x3F8

#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>


struct bruger_info
{
  char STbruger[40];
  char STkortnummer[40];
  char STkode[40];
  char STlog[40];
  struct bruger_info *naeste;
};

struct bruger_info *foerste, *sidste, *ny;
char streng[10];
int login = 0;                    //bruges til loginstatus

void delay(int wait)
{
  long int n,t;
  for(n=1; n<wait;n++){
    for(t=0;t<60000; t++){}
  }
}

int antallog() {
  char log[10] = {'1'};
  struct bruger_info *ptr = foerste;
```

```
  while(ptr != NULL) {
    if(strcmp(ptr->STlog, log) == 0) {
         printf("\nNogen er logget ind\n");
      login = 1;
         return 0;
    }
  ptr = ptr->naeste;
  }
    printf("\nIngen er logget ind\n");
    login = 0;
    return 0;
}

int skriv_til_log(char *beskrivelse, char *addlog) {
  FILE *fptr;
  struct time t;
   struct date d;
  gettime(&t);
   getdate(&d);

  if((fptr = fopen("log.txt", "a")) == NULL) {
    printf("\nFilen 'log.txt' kan ikke åbnes for tilføjelser.");
    return 0;
  }
  fprintf(fptr, "%2d-%02d-%02d %2d:%02d:%02d", d.da_day, d.da_mon, d.da_year, t.ti_hour, t.ti_min,
t.ti_sec);
  fprintf(fptr, " ");
  fputs(beskrivelse, fptr);
  fputs(addlog, fptr);
  fputs("\n", fptr);
  fclose(fptr);
  return 0;
}

int start() {
  while(1) {
    hentbruger();  //Opdaterer brugerstruct
    antallog();  //Tjekker om der er nogen logget ind(Lås on/off?)
```

```
    display(1);  //Viser standard-teksten
    laes_kort();
  }
}
```

```
int hentbruger()
{

  FILE *fptr;
  char ch;
  char linie[81];
  char s_temp[81];
  int j = 0, i = 0;

  foerste = NULL;

  if((fptr = fopen("brug.txt", "r")) == NULL) {
    printf("\nFilen brug.txt kan ikke åbnes.");
    skriv_til_log("Program exit with error code ", "Kunne ikke finde brug.txt");
    exit(1);
  }
  while(fgets(linie, 81, fptr) != NULL) {
    ny = malloc(sizeof(struct bruger_info));
    if(foerste == NULL)
      foerste = sidste = ny;
    else {
      sidste->naeste= ny;
      sidste = ny;
  }

  while((ch = linie[i++]) != ':') //Brugernavn
    s_temp[j++] = ch;

  s_temp[j] = '\0';
```

```
  strcpy(sidste->STbruger, s_temp);
  j = 0;

  while((ch = linie[i++]) != ':') //Kortnummer
    s_temp[j++] = ch;

  s_temp[j] = '\0';
  strcpy(sidste->STkortnummer, s_temp);
  j = 0;

  while((ch = linie[i++]) != ':') //Kode
    s_temp[j++] = ch;

  s_temp[j] = '\0';
  strcpy(sidste->STkode, s_temp);
  j = 0;

  while((ch = linie[i++]) != ':') //Login Status
    s_temp[j++] = ch;

  s_temp[j] = '\0';
  strcpy(sidste->STlog, s_temp);
  j = i = 0;
  sidste->naeste = NULL;

  }
    fclose(fptr);
    return 0;
}

int laeskode() {
  int ch;
  int pa_data, pb_data;
  int t_temp[9999];
  char kode[10];
  char tom[10] = {'\0'};
  int tal;
  strcpy(streng, tom);
```

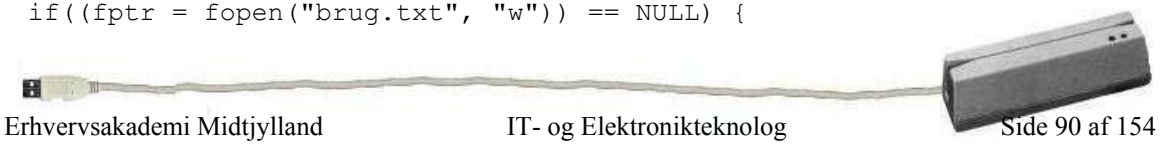
```
while(strlen(streng) < 4) {
  outportb(BASE + PB ,0x80);// Skriver B'10000000' til I/O kortet port B
  pa_data = inportb(BASE + PA); // Læser fra PORTA

  if(pa_data == 16) {
    tal = 1;
    itoa(tal, kode, 10);
    strcat(streng, kode);
    delay(50);
  }
  if(pa_data == 32) {
    tal = 4;
    itoa(tal, kode, 10);
    strcat(streng, kode);
    delay(50);
  }

  if(pa_data == 64) {
    tal = 7;
    itoa(tal, kode, 10);
    strcat(streng, kode);
    delay(50);
  }
  if(pa_data == 128) {
    tal = 1;
    itoa(tal, kode, 10);
    strcat(streng, kode);
    delay(50);
  }

  outportb(BASE + PB ,0x40);// Skriver B'01000000' til I/O kortet port B
  pa_data = inportb(BASE + PA); // Læser fra PORTA

  if(pa_data == 16) {
    tal = 2;
    itoa(tal, kode, 10);
    strcat(streng, kode);
```

```
    delay(50);
   }
   if(pa_data == 32) {
    tal = 5;
    itoa(tal, kode, 10);
    strcat(streng, kode);
    delay(50);
   }
   if(pa_data == 64) {
    tal = 8;
    itoa(tal, kode, 10);
    strcat(streng, kode);
    delay(50);
   }
   if(pa_data == 128) {
    tal = 0;
    itoa(tal, kode, 10);
    strcat(streng, kode);
    delay(50);
   }


outportb(BASE + PB ,0x20);// Skriver B'00100000' til I/O kortet port B
   pa_data = inportb(BASE + PA); // Læser fra PORTA

   if(pa_data == 16) {
    tal = 3;
    itoa(tal, kode, 10);
    strcat(streng, kode);
    delay(50);
   }
   if(pa_data == 32) {
    tal = 6;
    itoa(tal, kode, 10);
    strcat(streng, kode);
    delay(50);
   }
   if(pa_data == 64) {
```

```
     tal = 9;
     itoa(tal, kode, 10);
     strcat(streng, kode);
     delay(50);
    }
    if(pa_data == 128) {
     printf("#");
     delay(50);
    }
  }
   return;
}

int laes_kort() {
  struct bruger_info *ptr = foerste;
  int fundet = 0;
  char s_temp[81];
  char temp[81] = {'1'};
  int c, j = 0;
  char ch,tegn;
  outportb(PORT1 + 1 , 0);   /* Turn off interrupts - Port1 */
  outportb(PORT1 + 3 , 0x80);  /* SET DLAB ON */
  outportb(PORT1 + 0 , 0x0C);  /* Set Baud rate - Divisor Latch Low Byte */
  outportb(PORT1 + 1 , 0x00);  /* Set Baud rate - Divisor Latch High Byte */
  outportb(PORT1 + 3 , 0x03);  /* 8 Bits, No Parity, 1 Stop Bit */
  outportb(PORT1 + 2 , 0xC7);  /* FIFO Control Register */
  printf("Indlæs kort\n");
  do {
    c = inportb(PORT1 + 5);
    if (c & 1) {
      ch = inportb(PORT1);
      s_temp[j++] = ch;
    }
    if(kbhit()) {
      tegn = getch();
    }
    if(tegn == 27) {
      exit(0);
```

```
  }
 }
 while (ch !='?');
  s_temp[j] = '\0';
  while(ptr != NULL) {
   if(strcmp(ptr->STkortnummer, s_temp) == 0) {
    fundet = 1;
    printf("Kort fundet\n");
    break;
   }
   else
    ptr = ptr->naeste;
   }
   if(fundet) {
    if(strcmp(temp, ptr->STlog) == 0) {
     display(3);
     login_out(ptr->STkortnummer);
     printf("Bruger er logget ud\n");
     skriv_til_log("Users logout :  ", ptr->STbruger );
     return 0;
    }
    else {
     dioder(1);
     delay(100);
     printf("Indtast kode\n");
     laeskode();
     if(strcmp(streng, ptr->STkode) == 0) {
      login_out(ptr->STkortnummer);
      printf("Bruger er logget ind\n");
      skriv_til_log("Users login :  ", ptr->STbruger );
      display(2);
      return 0;
    }
    else {
     printf("Følgende bruger har prøvet at logge ind, med fejl i koden: %s\n", ptr->STbruger);
     skriv_til_log("Følgende bruger har prøvet at logge ind, med fejl i koden: ", ptr->STbruger );
     display(4);
     return 0;
```

```
      }
     }
    }
    else {
     printf("Følgende kort har prøvet at logge ind, med fejl: %s\n", s_temp);
     skriv_til_log("Følgende kort har prøvet at logge ind, med fejl: ", s_temp );
     display(4);
     return 0;
    }
}

int login_out(char *kortnummer) {

  struct bruger_info *ptr = foerste;

  while(ptr != NULL) {
   if(strcmp(ptr->STkortnummer, kortnummer) == 0) {
    opdatefil(kortnummer);
    return;
   }
   else
    ptr = ptr->naeste;
  }
  return;
}

int opdatefil(char *kortnummer[81])
{
  char VAnavn[81];
  char ch;
  int i = 0;
  int s_temp[81];
  int VAkortnummer, VAkode;
  FILE *fptr;
  char temp[10] = {'0'};
  struct bruger_info *ptr = foerste;

  if((fptr = fopen("brug.txt", "w")) == NULL) {
```

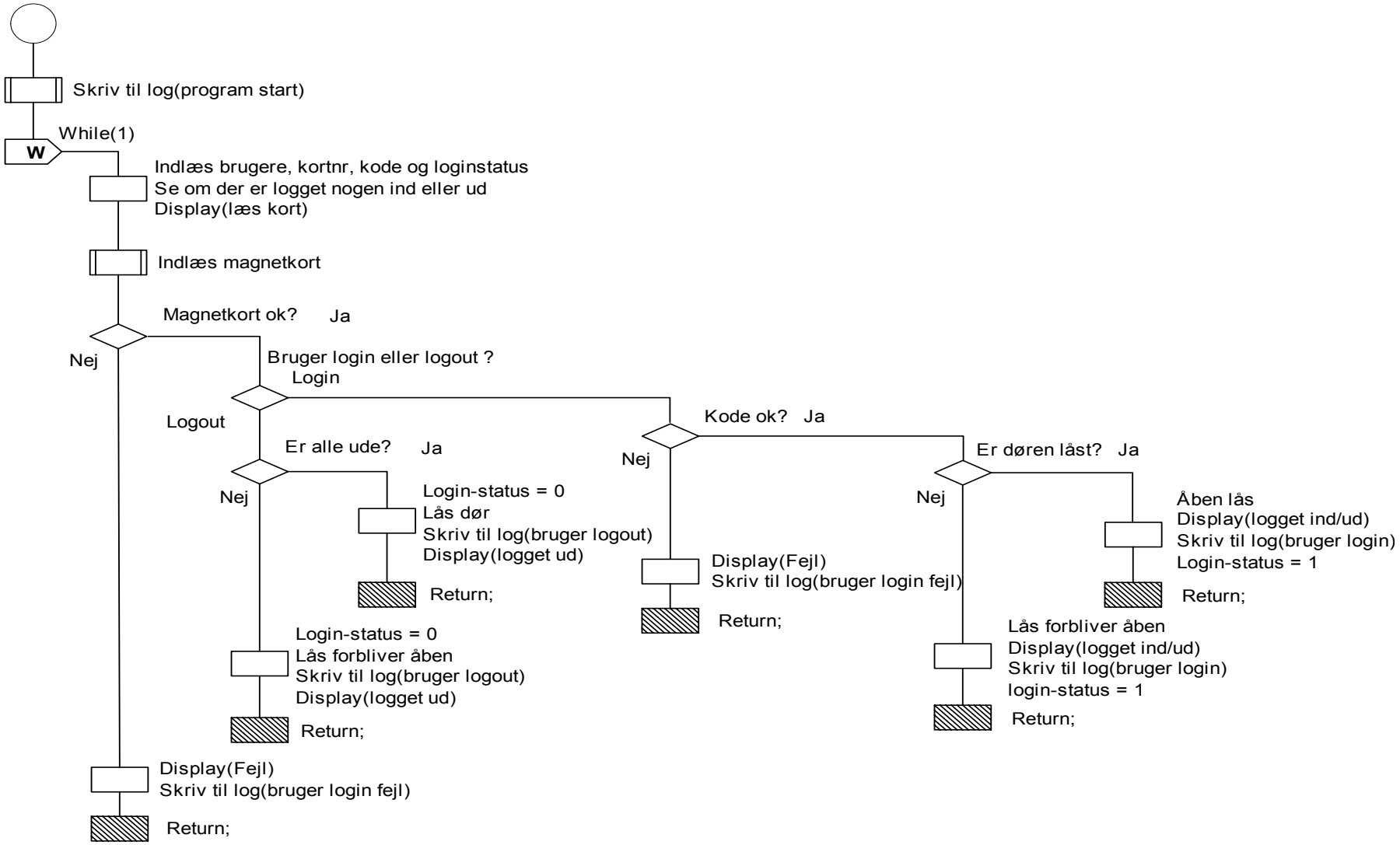


```
      printf("Filen brug.txt kunne ikke åbnes");
   skriv_til_log("Programfejl -> ", "Kunne ikke finde brug.txt");
      exit(1);
   }
while(ptr != NULL) {
     if(i++ != 0)
      putc('\n', fptr);
      fputs(ptr->STbruger, fptr);
      putc(':', fptr);
      fprintf(fptr, "%s", ptr->STkortnummer);
      putc(':', fptr);
      fprintf(fptr, "%s", ptr->STkode);
      putc(':', fptr);
      if(strcmp(ptr->STkortnummer, kortnummer) == 0) {
        if(strcmp(ptr->STlog, temp) == 0)
          fprintf(fptr, "1");
        else
          fprintf(fptr, "0");
     }
     else {
      fprintf(fptr, "%s", ptr->STlog);
     }
     putc(':', fptr);
     ptr = ptr->naeste;
}
  fclose(fptr);
  return;
}
```

```
int dioder(int farve) {

  if(farve == 0) {                                                    // Grøn diode
    outportb(BASE + PB ,0x10);              // Skriver til binær-output 16 på Port B
  }
  if(farve == 1) {                                                    // Rød diode
    outportb(BASE + PB ,0x08);              // Skriver til binær-output 8 på Port B
  }
}

int display(int valg) {
   int pc_data;

   if(valg == 1) {                          //standard
     outportb(BASE + PC ,0x00 + login);     //Skriver 0000000 ud på I/O-kortet = "Laes kort/Tast kode!!"
   }
   if(valg == 2) {                          //login
     outportb(BASE + PC ,0x80 + login);     //Skriver 1000000 ud på I/O-kortet = "Du er nu logget ind!"
     dioder(1);                                                       //Den grønne diode tændes
   }
   if(valg == 3) {                                    //logout
     outportb(BASE + PC ,0x40 + login);     //Skriver 0100000 ud på I/O-kortet = "Du er nu logget ud!"
     dioder(1);                                                       //Den grønne diode tændes

   }
   if(valg == 4) {                                    //fejl
     outportb(BASE + PC ,0xC0 + login);     //Skriver 1100000 ud på I/O-kortet = "Forkert kort/kode!!"
     dioder(0);                                                       //Den røde diode tændes
   }
   if(valg != 1)
     delay(500);
     outportb(BASE + PB ,0x00);             // Slukker dioder efter brug.
}
int main() {
  outportb(BASE + 3,CWD); //index to CH0 CWD sætter
  skriv_til_log("Start ", "login Server");
  start();
}
```

Bilag 4

-Flowdiagram til loginstyring


Skriv til log(program start)
While(1)
W
Indlæs brugere, kortnr, kode og loginstatus
Se om der er logget nogen ind eller ud
Display(læs kort)
Indlæs magnetkort
Magnetkort ok? Ja
Nej
Bruger login eller logout ?
Login
Logout
Er alle ude? Ja
Nej
Login-status = 0
Lås dør
Skriv til log(bruger logout)
Display(logget ud)
Return;
Login-status = 0
Lås forbliver åben
Skriv til log(bruger logout)
Display(logget ud)
Return;
Kode ok? Ja
Nej
Display(Fejl)
Skriv til log(bruger login fejl)
Return;
Er døren låst? Ja
Nej
Åben lås
Display(logget ind/ud)
Skriv til log(bruger login)
Login-status = 1
Return;
Lås forbliver åben
Display(logget ind/ud)
Skriv til log(bruger login)
login-status = 1
Return;
Display(Fejl)
Skriv til log(bruger login fejl)
Return;

Bilag 5

-Jinghua JM204A LCD

● AC Characteristics (Vdd=4.5V~5.5V,Ta=-30~+85℃)

Mode	Characteristic	Symbol	Min.	Typ.	Max.	Unit
Write Mode	E Cycle Time	t_C	500	-	-	ns
	E Rise/Fall Time	t_R,t_F	-	-	20	
	E Pulse Width (High,Low)	t_W	230	-	-	
	R/W and RS Setup Time	t_{SU1}	40	-	-	
	R/W and RS Hold Time	t_{H1}	10	-	-	
	Data Setup Time	t_{SU2}	80	-	-	
	Data Hold Time	t_{H2}	10	-	-	
Read Mode	E Cycle Time	t_C	500	-	-	ns
	E Rise/Fall Time	t_R,t_F	-	-	20	
	E Pulse Width (High,Low)	t_W	230	-	-	
	R/W and RS Setup Time	t_{SU}	40	-	-	
	R/W and RS Hold Time	t_H	10	-	-	
	Data Output Delay Time	t_D	-	-	120	
	Data Hold Time	t_{DH}	5	-	-	

● IC Specifications

See The Reference of Samsung Data Book-----S6A0700(KS0070B)

● Pin assignment

Pin NO.	Symbol	Function		Remark
1	GND	Power supply	0V	
2	Vdd		+5V	
3	V5		For LCD	Variable
4	RS	Register Select(H=Data,L=Instruction)		
5	R/W	Read/Write L=MPU to LCM,H=LCM to MPU		
6	E	Enable		
7	DB0	Data bus bit 0		
8	DB1	Data bus bit 1		
9	DB2	Data bus bit 2		
10	DB3	Data bus bit 3		
11	DB4	Data bus bit 4		
12	DB5	Data bus bit 5		
13	DB6	Data bus bit 6		
14	DB7	Data bus bit 7		
15	A	Anode of LED Unit		
16	K	Cathode of LED Unit		

● Reflector of Screen and DDRAM Address

Display position	1-1	1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9	1-10
DDRAM address	00	01	02	03	04	05	06	07	08	09
Display position	1-11	1-12	1-13	1-14	1-15	1-16	1-17	1-18	1-19	1-20
DDRAM address	0A	0B	0C	0D	0E	0F	10	11	12	13
Display position	3-1	3-2	3-3	3-4	3-5	3-6	3-7	3-8	3-9	3-10
DDRAM address	14	15	16	17	18	19	1A	1B	1C	1D
Display position	3-11	3-12	3-13	3-14	3-15	3-16	3-17	3-18	3-19	3-20
DDRAM address	1E	1F	20	21	22	23	24	25	26	27
Display position	2-1	2-2	2-3	2-4	2-5	2-6	2-7	2-8	2-9	2-10
DDRAM address	40	41	42	43	44	45	46	47	48	49
Display position	2-11	2-12	2-13	2-14	2-15	2-16	2-17	2-18	2-19	2-20
DDRAM address	4A	4B	4C	4D	4E	4F	50	51	52	53
Display position	4-1	4-2	4-3	4-4	4-5	4-6	4-7	4-8	4-9	4-10
DDRAM address	54	55	56	57	58	59	5A	5B	5C	5D
Display position	4-11	4-12	4-13	4-14	4-15	4-16	4-17	4-18	4-19	4-20
DDRAM address	5E	5F	60	61	62	63	64	65	66	67

"1-1" means first character of line 1 on screen

● Instruction Table

Instruction	Instruction Code										Description	Execution Time(fosc= 270kHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write"20H" to DDRAM set DDRAM address to "00H" from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed	1.53ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display	39 μs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Set display(D) cursor(C) and blinking of cursor(B) on/off	39 μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display shift control bit,and the direction, without changing DDRAM data	39 μs
Function Set	0	0	0	0	1	DL	N	F	-	-	Set interface data length(DL:8bit/4bit), number of display line (N:2line/1line) and,display font type F:5X11dots / 5X8dots	39 μs
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	39 μs
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	39 μs
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF The contents of address counter can also be read	0 μs
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM)	43 μs
Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM)	43 μs

● Instruction Description

A. Clear Display

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

Clear all the display data by writing "20H"(space code) to all DDRAM address,and set DDRAM address to "00H" into AC(address counter).
Return cursor to the original status,namely,bring the cursor to the left edge on the first line of the display.
Make the entry mode increment(I/D="High").

B. Return Home

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	-

Set DDRAM address to "00H" into the address counter.
Return cursor to its original site and return display to its original status,if shifted.
Contents of DDRAM does not change.

C. Entry Mode Set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	SH

Set the moving direction of cursor and display.

I/D:Increment /decrement of DDRAM address(cursor or blink)

I/D=High,cursor/blink moves to right and DDRAM address is increased by 1.
I/D=low,cursor/blink moves to left and DDRAM address is decreased by 1.

*CGRAM operates the same way as DDRAM, when reading from or writing to CGRAM.

SH:Shift of entire display

When DDRAM read (CGRAM read/write) operation or SH=Low,shifting of entire display is not performed.if SH=High, and DDRAM write operation,shift of entire display is performed according to I/D value(I/D=High,shift left,I/D=Low, shift right).

D. Display ON/OFF Control

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

D:Display ON/OFF control bit

When D=High, entire display is turned on.

When D=Low, display is turned off,but display data remains in DDRAM.

C:Cursor ON/OFF control bit

When C=High, cursor is turned on.

When C=Low, cursor is disappeared in current display ,but I/D register preserves its data.

B:Cursor Blink ON/OFF control bit

When B=High, cursor blink is on, which performs alternately between all the "High" data and display characters at the cursor position.

When B=Low ,blink is off.

E. Cursor or Display Shift

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	-	-

Shifting of right/left cursor position or display without writing or reading of display data.

This instruction is used to correct or search display data.

During 2-line mode display,cursor moves to the 2^{nd} line after the 40^{th} digit of the 1^{st} line.

Note that display shift is performed simultaneously in all the lines.

When displayed data is shifted repeatedly,each line is shifted individually.

When display shift is performed,the contents of the address counter are not changed.

S/C	R/L	Operation
0	0	Shift cursor to the left, AC is decreased by1
0	1	Shift cursor to the right, AC is increased by1
1	0	Shift all the display to the left,cursor moves according to the display
1	1	Shift all the display to the right,cursor moves according to the display

F. Function set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	-	-

DL:Interface data length control bit
When DL=High, it means 8-bit bus mode with MPU.
When DL=Low, it means 4-bit bus mode with MPU.
When 4-bit bus mode, it needs to transfer 4-bit data twice.
N:Display line number control bit
When N=Low, 1-line display mode is set.
When N=High, 2-line display mode is set.
F:Display font type control bit
When F=Low, 5x8 dots format display mode is set.
When F=High, 5x11 dots format display mode.

G. Set CGRAM Address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

Set CGRAM address to AC.
This instruction makes CGRAM data available from MPU.

H. Set DDRAM Address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0

Set DDRAM address to AC.
This instruction makes DDRAM data available from MPU.
When 1-line display mode(N=Low),DDRAM address is from “00H” to “4FH”.
In 2-line display mode(N=High),DDRAM address in the 1st line is from “00H” to “27H”,and DDRAM address in the 2nd line is from “40H” to “67H”.

I. Read Busy Flag & Address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0

This instruction shows whether IC is in internal operation or not .
If BF is "High",internal operation is in progress and should wait until BF is to be Low,which by then the next instruction can be performed. In this instruction you can also read the value of the address counter.

J. Write data to RAM

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D7	D6	D5	D4	D3	D2	D1	D0

Write binary 8-bit data to DDRAM/CGRAM.
The selection of RAM from DDRAM,and CGRAM,is set by the previous address set instruction(DDRAM address set,CGRAM address set).
RAM set instruction can also determine the AC direction to RAM.
After write operation, the address is automatically increased /decreased by 1,according the entry mode.

K. Read data from RAM

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D7	D6	D5	D4	D3	D2	D1	D0

Read binary 8-bit data from DDRAM/CGRAM.
The selection of RAM is set by the previous address set instruction.If the address set instruction of RAM is not performed before this instruction, the data that has been read first is invalid, as the direction of AC is not yet determined. If RAM data is read several times without RAM address instructions set before read operation,the correct RAM data can be obtained from the second. But the first data would be incorrect,as there is no time margin to transfer RAM data.

Bilag 6

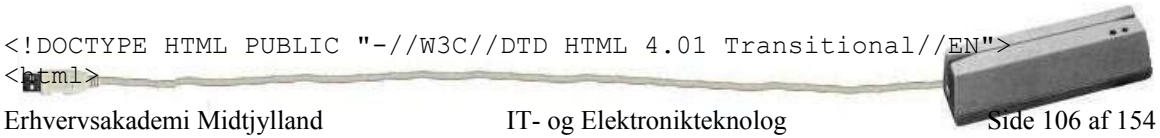
-Tidsplan

Bilag 7

-Kildekode til Hjemmeside

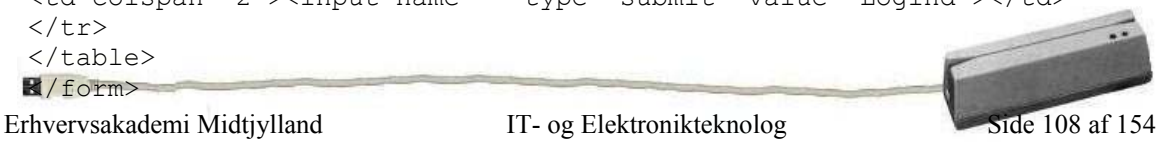
Default.php

```
<?
//
//   default.php
//
//  Lavet af Ole Rosengreen & Kenneth Dalbjerg
//
session_start();
include("includes/function.php");
$action = $_POST["action"];
if($action == "") {
  $action = $_GET["action"];
}

if($action == "logud") {
// tjekker for om $action = logud
  $_SESSION["login"] = "nej";
  //Sætter session
  $action = "";
}

if($action == "logind") {
$brugernavn = $_POST["brugernavn"];
$password = $_POST["password"];
  if($brugernavn == "" OR $password == "") {
    echo "Du skal indtaste både brugernavn & password, for at kunne logge ind.";
    $action = "";
   } else {
    if($brugernavn == "admin" AND $password == "test") {
      $_SESSION["login"] = "ja";
     } else {
      $_SESSION["login"] = "nej";
      echo "Forkert brugernavn eller kodeord";
      $action = "";
    }
  }
 } ?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
```



```
<head>
<title>Administration</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<SCRIPT LANGUAGE="JavaScript">
<!--
 function formCheck ()
 {

  if (document.Logind.brugernavn.value == "")
  {
    alert("Skriv venligst et brugernavn");
    document.Logind.brugernavn.focus();
    return false;
  }

  if (document.Logind.brugernavn.value == "Indtast dit brugernavn her")
  {
    alert("Skriv venligst et brugernavn");
    document.Logind.brugernavn.focus();
    return false;
  }

  if (document.Logind.password.value == "")
  {
    alert("Skriv venligst et password");
    document.Logind.password.focus();
    return false;
  }
}
// -->
</SCRIPT>
</head>
<body marginwidth="0" marginheight="0" leftmargin="0" topmargin="0">
<table width="100%" height="100%" border="1" bordercolor="#000000">
<tr>
<td width="17%" valign="top" bgcolor="#D0DCE0">
<? menu(); ?>
</td>
<td width="83%" valign="top" bgColor="#f5f5f5">
<?if($tekst != "") {?>
  <font color="#FF0000"><? echo $tekst ?></font><br><br>
<? } ?>
```

```
<? if($action == "vislog") {
  if($_SESSION["login"] == "ja") {
    vislog();
     } else {?>
    Du skal være logget ind
  <? }
}
if($action == "vislogged") {
  if($_SESSION["login"] == "ja") {
    loggedinuser();
   } else {?>
    Du skal være logget ind
  <? }
}

if($action == "logind") { ?>
  Du er nu logget ind.
<? }
if($action == "") { ?>
  Indtast dit brugernavn & password herunder
  <form name="Logind" action="default.php" method="post" onSubmit="return formCheck();">
  <input name="action" type="hidden" value="logind">
  <table width="30%" border="0">
  <tr>
  <td>Brugernavn</td>
  <td>
  <input name="brugernavn" type="text" value="Indtast dit brugernavn her" size="40" onFocus="value=''">
  </td>
  </tr>
  <tr>
  <td>Password</td>
  <td><input name="password" type="password" size="40"></td>
  </tr>
  <tr>
  <td colspan="2"><input name="" type="submit" value="Logind"></td>
  </tr>
  </table>
  </form>
```



```
<? } ?>
</td>
</tr>
</table>
</body>
</html>
```

Funktion.php

```
<? function vislog() {
  echo "Log filen:<br>";
  $content = file("c:\projekt\log.txt") or die ("Kunne ikke åbne log.txt");
  for ($i=0; $i<sizeof($content); $i++) {
    echo htmlentities("$content[$i]");
    echo "<br>";
  }
}

function loggedinuser() {
  echo "Følgende bruger er logget ind:<br>";
  $content = file("c:\projekt\brug.txt") or die ("Kunne ikke åbne brug.txt");
  for ($i=0; $i<sizeof($content); $i++) {
    $temp = explode(":", trim($content[$i]) );
    if ($temp[3] == 1) { echo "$temp[0]<br>"; }
  }
}

function menu() {
  if($_SESSION["login"] == "ja") {
    echo "<a href=\"default.php?action=vislogged\">Hvem er logget ind</a><br>";
    echo "<a href=\"default.php?action=vislog\">Vis loggen</a><br>";
    echo "<a href=\"default.php?action=logud\">Log ud</a><br>";
  } else {
    echo "<a href=\"default.php\">Log ind</a><br>";
  }
}
?>
```

Bilag 8

-Assemblerkode til PIC16C84A

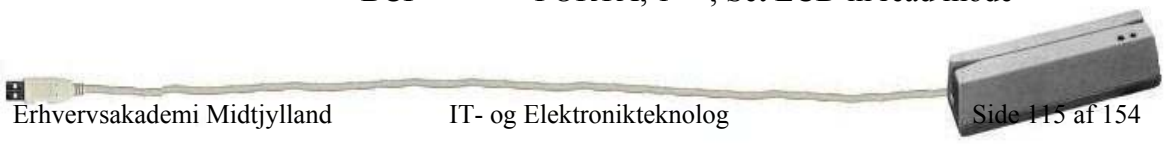
```
                LIST P=16F84A

                include p16F84A.inc

;*****************************************************************
*******
; Initialisering af variabler:
;*****************************************************************
*******
LCD_TEMP EQU    0x020                   ; LCD subroutines internal use
TEKST_TEMP      EQU     0x021           ; Index to table
strings
WCYCLE          equ     0x023
;*****************************************************************
*******

                ORG     0x0000          ; RESET vector
location
RESET           GOTO    START

                ORG     0x0004          ; Interrupt vector
location
                GOTO    START

#include "bank.inc"
#include "wait.inc"

;*****************************************************************
*******
; Definering af processor registre:
;*****************************************************************
*******
START
                CLRF    STATUS          ; Do initialization,
Select bank 0
                CLRF    PORTA           ; ALL PORT output
should output Low.
                CLRF    PORTB

                BANK1
                MOVLW   0x0F8           ; RA2-0 outputs,
RA4-3 inputs
                MOVWF   TRISA
                MOVLW   0x000           ; RB7-0 outputs
                MOVWF   TRISB
                BANK0
```

```
;*****************************************************************************
*******
; Main:
;*****************************************************************************
*******
                        CALL    LCDINIT         ; Initialize
LCDisplay
standard:
                        CALL    LCDCLEAR
                        CALL    TEKST1          ; Display message
menu:
                        BTFSC   PORTA,4
                        CAll    login

                        BTFSC   PORTA,3
                        CALL    logout

                        goto    menu
login:
                        CALL    LCDCLEAR
                        BTFSC   PORTA,3
                        CALL    fejl

                        CALL    TEKST2

login2:                 BTFSS   PORTA,4
                        CALL    LCDCLEAR
                        BTFSS   PORTA,4
                        CALL    standard
                        goto    login2

logout:
                        CALL    LCDCLEAR
                        BTFSC   PORTA,4
                        CALL    fejl
                        CALL    TEKST4

logout2:    BTFSS       PORTA,3
                        CALL    LCDCLEAR
                        BTFSS   PORTA,3
                        CALL    standard
                        goto    logout2

fejl:
                        CALL    TEKST3
fejl2:                  BTFSC   PORTA,4
                        goto    fejl2
                        goto    standard
```

```
;*****************************************************************
*******
; Tekst-styring: (tekst 1+2+3+4)
;*****************************************************************
*******

TEKST1
                        MOVLW   0                       ; Startindex of table
message
DISP_MSG
                        MOVWF   TEKST_TEMP              ; Holds message
address
                        CALL    MSG1
                        ANDLW   0x0FF                   ; Check if at end of
message
                        BTFSC   STATUS, Z  ; (zero returned at end)
                        GOTO    TEKST1_SLUT
                        CALL    LCDPUTCHAR              ; Display character
                        MOVF    TEKST_TEMP, W           ; Point to next
character
                        ADDLW   1
                        GOTO    DISP_MSG
TEKST1_SLUT             RETURN

TEKST2
                        MOVLW   0                       ; Startindex of table
message
DISP_MSG2
                        MOVWF   TEKST_TEMP              ; Holds message
address
                        CALL    MSG2
                        ANDLW   0x0FF                   ; Check if at end of
message
                        BTFSC   STATUS, Z  ; (zero returned at end)
                        GOTO    TEKST2_SLUT
                        CALL    LCDPUTCHAR              ; Display character
                        MOVF    TEKST_TEMP, W           ; Point to next
character
                        ADDLW   1
                        GOTO    DISP_MSG2
TEKST2_SLUT             RETURN

TEKST3
                        MOVLW   0                       ; Startindex of table
message
DISP_MSG3
```

```
                        MOVWF   TEKST_TEMP              ; Holds message
address
                        CALL    MSG3
                        ANDLW   0x0FF                   ; Check if at end of
message
                        BTFSC   STATUS, Z  ; (zero returned at end)
                        GOTO    TEKST3_SLUT
                        CALL    LCDPUTCHAR              ; Display character
                        MOVF    TEKST_TEMP, W           ; Point to next
character
                        ADDLW   1
                        GOTO    DISP_MSG3
TEKST3_SLUT             RETURN

TEKST4
                        MOVLW   0                       ; Startindex of table
message
DISP_MSG4
                        MOVWF   TEKST_TEMP              ; Holds message
address
                        CALL    MSG4
                        ANDLW   0x0FF                   ; Check if at end of
message
                        BTFSC   STATUS, Z  ; (zero returned at end)
                        GOTO    TEKST4_SLUT
                        CALL    LCDPUTCHAR              ; Display character
                        MOVF    TEKST_TEMP, W           ; Point to next
character
                        ADDLW   0x001
                        GOTO    DISP_MSG4
TEKST4_SLUT             RETURN
;*****************************************************************
*******
; Initialisering og opsætning af display
;*****************************************************************
*******

LCDINIT

                        CLRF    PORTA                   ; ALL PORT output
should output Low.
                        WAIT    .100                    ; Venter 10
millisekunder

                        MOVLW   0x038                   ; 8-bit-interface, 2-
lines
                        CALL    LCDPUTCMD
```

```
                MOVLW   0x000                   ; disp.off, curs.off,
no-blink
                CALL    LCDCURSOR
                CALL    LCDCLEAR
                MOVLW   0x004                   ; disp.on, curs.off
                CALL    LCDCURSOR
                RETURN

LCDBUSY
                BANK1
                MOVLW   0x0FF                   ; Set PORTB for
input
                MOVWF   TRISB
                BANK0
                BCF     PORTA, 0    ; Setup to read busy flag
                BSF     PORTA, 1    ; READ/WRITE
                BSF     PORTA, 2    ; LCD E-line High
                MOVF    PORTB, W    ; Read busy flag + DDram
address
                BCF     PORTA, 2    ; LCD E-line Low
                ANDLW   0x80                    ; Check Busy flag,
High = Busy
                BTFSS   STATUS, Z
                GOTO    LCDBUSY
LCDNOTBUSY      BCF     PORTA, 1
                BANK1
                MOVLW   0x000
                MOVWF   TRISB                   ; Set PORTB for
output
                BANK0
                RETURN

LCDCLEAR
                MOVLW   0x001
                CALL    LCDPUTCMD
                RETURN

LCDCURSOR
                ANDLW   0x007                   ; Strip upper bits
                IORLW   0x008                   ; Function set
                CALL    LCDPUTCMD
                RETURN

LCDPUTCHAR
                MOVWF   LCD_TEMP ; Character to be sent is in W
                CALL    LCDBUSY                 ; Wait for LCD to
be ready
                BCF     PORTA, 1    ; Set LCD in read mode
```



```
                BSF     PORTA, 0    ; Set LCD in data mode
                BSF     PORTA, 2    ; LCD E-line High
                MOVF    LCD_TEMP, W
                MOVWF   PORTB               ; Send data to LCD
                BCF     PORTA, 2    ; LCD E-line Low
                RETURN

LCDPUTCMD
                MOVWF   LCD_TEMP ; Command to be sent is in W
                CALL    LCDBUSY             ; Wait for LCD to
be ready
                BCF     PORTA, 1    ; Set LCD in read mode
                BCF     PORTA, 0    ; Set LCD in command mode
                BSF     PORTA, 2    ; LCD E-line High
                MOVF    LCD_TEMP, W
                MOVWF   PORTB               ; Send data to LCD
                BCF     PORTA, 2    ; LCD E-line Low
                RETURN

MSG1
                addwf   PCL ,F              ;Jump to char
pointed to in W reg
                retlw   'L'
                retlw   'a'
                retlw   'e'
                retlw   's'
                retlw   ' '
                retlw   'k'
                retlw   'o'
                retlw   'r'
                retlw   't'
                retlw   '/'
                retlw   'T'
                retlw   'a'
                retlw   's'
                retlw   't'
                retlw   ' '
                retlw   'k'
                retlw   'o'
                retlw   'd'
                retlw   'e'
                retlw   '!'
MSG1_END
                retlw   0

MSG2
                addwf   PCL ,F              ;Jump to char
pointed to in W reg
```

```
                retlw   'D'
                retlw   'u'
                retlw   ' '
                retlw   'e'
                retlw   'r'
                retlw   ' '
                retlw   'n'
                retlw   'u'
                retlw   ' '
                retlw   'l'
                retlw   'o'
                retlw   'g'
                retlw   'g'
                retlw   'e'
                retlw   't'
                retlw   ' '
                retlw   'i'
                retlw   'n'
                retlw   'd'
                retlw   '!'
MSG2_END
                retlw   0
MSG3
                addwf   PCL ,F                  ;Jump to char
pointed to in W reg
                retlw   'F'
                retlw   'o'
                retlw   'r'
                retlw   'k'
                retlw   'e'
                retlw   'r'
                retlw   't'
                retlw   ' '
                retlw   'k'
                retlw   'o'
                retlw   'r'
                retlw   't'
                retlw   '/'
                retlw   'k'
                retlw   'o'
                retlw   'd'
                retlw   'e'
                retlw   '!'
                retlw   '!'
                retlw   ' '
MSG3_END
                retlw   0
```

```
MSG4
                addwf   PCL ,F          ;Jump to char
pointed to in W reg
                retlw   'D'
                retlw   'u'
                retlw   ' '
                retlw   'e'
                retlw   'r'
                retlw   ' '
                retlw   'n'
                retlw   'u'
                retlw   ' '
                retlw   'l'
                retlw   'o'
                retlw   'g'
                retlw   'g'
                retlw   'e'
                retlw   't'
                retlw   ' '
                retlw   'u'
                retlw   'd'
                retlw   '!'
                retlw   ' '
MSG4_END
                retlw   0

        END                             ; End of program
```

Bilag 9

-Samlet diagram over kredsløbet

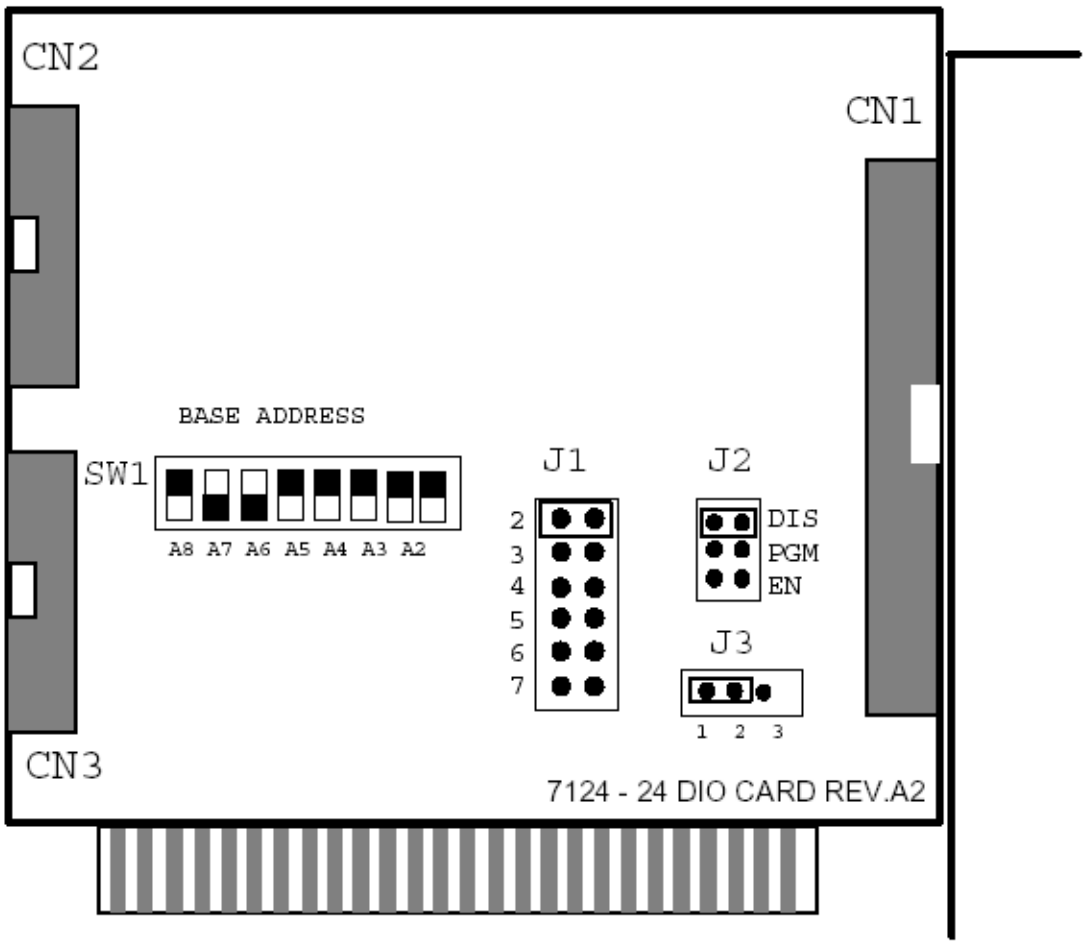
Bilag 10

-Ruko el-slutblik

Bilag 11

-I/O kort ACL-7124

I/O channels	24
Input Signal	Logic High Voltage :2.0 V to 5.25V Logic Low Voltage : 0.0 V to 0.80V Logic High Current : 20.0 uA Logic Low Current : -0.2 mA
Output Signal	Logic High Voltage : Minimum 2.4 V Logic Low Voltage : Maximum 0.5V Logic High Current : -15.0 mA Logic Low Current : 24.0 mA
Operating Temperature	0° ~ 60° C
Storage Temperature	-20° ~ 80° C
Humidity	5% ~ 95% non-condensing
I/O Connector	50-pin male ribbon cable connector
Bus	PC/XT Bus
IRQ Level	IRQ2 ~ IRQ7
I/O port address	4 bytes(Hex 200 ~ Hex 3FF)
Power Consumption	0.5A @5VDC (Typical) 0.8A @5VDC (Maximum)
Transfer Rate	300 K bytes/sec (Typical) 500 K bytes/sec (Maximum)
Size	(110mm X 97mm)


CN2
CN1
BASE ADDRESS
SW1
A8 A7 A6 A5 A4 A3 A2
J1
2
3
4
5
6
7
J2
DIS
PGM
EN
J3
1 2 3
CN3
7124 - 24 DIO CARD REV.A2

Jumper and DIP Switch Description

You can change the ACL-7124's base address and interrupt by setting DIP switches and jumpers on the card. The card's jumpers and switches are preset at the factory. Under normal circumstances, you should not need to change the jumper settings.

A jumper switch is closed (sometimes referred to as "shorted" with the plastic cap inserted over two pins of the jumper). A jumper is open with the plastic cap inserted over one or no pin(s) of the jumper.

Base Address Setting

The ACL-7124 requires 4 consecutive address locations in I/O address space. The base address of the ACL-7124 is restricted by the following conditions.

1. The base address must be within the range Hex 200 to Hex 3FF.

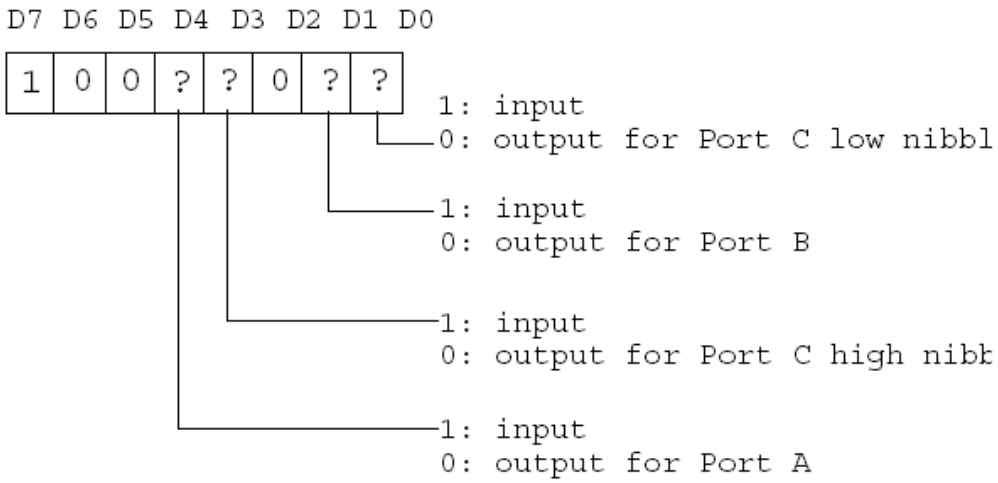
2. The base address should not conflict with any PC reserved I/O address. (refer to Appendix A)

The address setting of the ACL-7124 is described in Table 2.3.

Figure 2.3 Default DIP switch setting

I/O port address(hex)	1 A8	2 A7	3 A6	4 A5	5 A4	6 A3	7 A2	8 --
200-203	0 (ON)	1 (ON)	1 (ON)	0 (ON)	0 (ON)	0 (ON)	0 (ON)	X
: :								
(*) 2C0-2C3	0 (ON)	1 (OFF)	1 (OFF)	0 (ON)	0 (ON)	0 (ON)	0 (ON)	X
: :								
3F8-3FB	1 (OFF)	1 (OFF)	1 (OFF)	1 (OFF)	1 (OFF)	1 (OFF)	0 (ON)	X
3FC-3FF	1 (OFF)	1 (OFF)	1 (OFF)	1 (OFF)	1 (OFF)	1 (OFF)	1 (OFF)	X

(*) : default setting
X : Don't care
ON = 0 ; OFF = 1
A2, ..., A8 are corresponding to address lines of ISA bus.
A9 is always **1** (OFF).



Config. Value	D 4	D 3	D 1	D 0	PORT A	PORT C UPPER	PORT B	PORT C LOWER
80H	0	0	0	0	O/P	O/P	O/P	O/P
81H	0	0	0	1	O/P	O/P	O/P	I/P
82H	0	0	1	0	O/P	O/P	I/P	O/P
83H	0	0	1	1	O/P	O/P	I/P	I/P
88H	0	1	0	0	O/P	I/P	O/P	O/P
89H	0	1	0	1	O/P	I/P	O/P	I/P
8AH	0	1	1	0	O/P	I/P	I/P	O/P
8BH	0	1	1	1	O/P	I/P	I/P	I/P
90H	1	0	0	0	I/P	O/P	O/P	O/P
91H	1	0	0	1	I/P	O/P	O/P	I/P
92H	1	0	1	0	I/P	O/P	I/P	O/P
93H	1	0	1	1	I/P	O/P	I/P	I/P
98H	1	1	0	0	I/P	I/P	O/P	O/P
99H	1	1	0	1	I/P	I/P	O/P	I/P
9AH	1	1	1	0	I/P	I/P	I/P	O/P
9BH	1	1	1	1	I/P	I/P	I/P	I/P

I/O Address	Device
000-01F	DMA controller 1
020–03F	interrupt controller
040–05F	Timer
060–06F	Keyboard
070–07F	Real-time clock
080–09F	DMA page register
0A0–0BF	Interrupt controller 2
0C0–0DF	DMA controller
0F0–0FF	Math coprocessor
100–1EF	not usable
1F0–1F8	Fixed disk
200–207	Game I/O
278–27F	Parallel printer port 2 (LPT2:)
2F8–2FF	Serial Port 2 (COM2:)
300–31F	Prototype card
360–36F	Reserved
378–37F	Parallel printer port 1 (LPT1:)
3B0–3BF	Monochrome display
3C0–3CF	Reserved
3D0–3DF	Color graphics display
3F0–3F7	Diskette controller
3F8–3FF	Serial port 1 (COM 1:)

Bilag 12

-Microprocessor PIC16C84A

MICROCHIP

PIC16C84

8-bit CMOS EEPROM Microcontroller

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single cycle (400 ns @ 10 MHz) except for program branches which are two-cycle
- Operating speed: DC - 10 MHz clock input
  DC - 400 ns instruction cycle
- 14-bit wide instructions
- 8-bit wide data path
- 1K x 14 EEPROM program memory
- 36 x 8 general purpose registers (SRAM)
- 64 x 8 on-chip EEPROM data memory
- 15 special function hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt on change
  - Data EEPROM write complete
- 1,000,000 data memory EEPROM ERASE/WRITE cycles
- EEPROM Data Retention > 40 years

Peripheral Features:

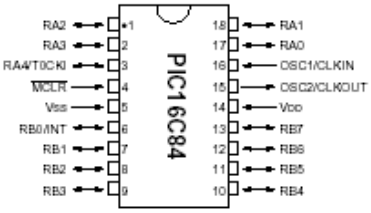
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 20 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

- Power-on Reset (POR)
- Power-up Timer (PWRT)
- Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Serial In-System Programming - via two pins

Pin Diagram

PDIP, SOIC



CMOS Technology:

- Low-power, high-speed CMOS EEPROM technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 6.0V
  - Industrial: 2.0V to 6.0V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 60 µA typical @ 2V, 32 kHz
  - 26 µA typical standby current @ 2V

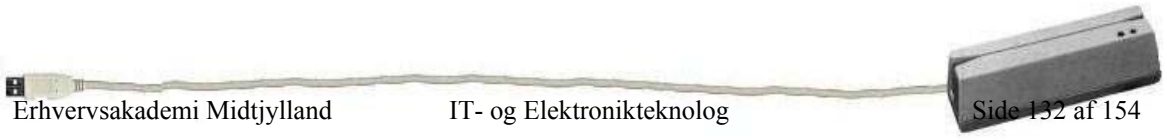
TABLE 9-2 PIC16CXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 1fff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.



Bilag 13

-Relæ SGR 462

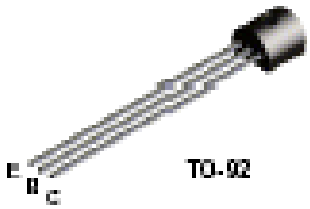
Bilag 14

-BC547

FAIRCHILD
SEMICONDUCTOR™

Discrete POWER & Signal Technologies

BC547
BC547A
BC547B
BC547C



TO-92

NPN General Purpose Amplifier

This device is designed for use as general purpose amplifiers and switches requiring collector currents to 300 mA. Sourced from Process 10. See PN100A for characteristics.

Absolute Maximum Ratings*

T_A = 25°C unless otherwise noted

Symbol	Parameter	Value	Units
V_{CEO}	Collector-Emitter Voltage	45	V
V_{CBO}	Collector-Base Voltage	50	V
V_{EBO}	Emitter-Base Voltage	6.0	V
I_C	Collector Current - Continuous	500	mA
T_J, T_{stg}	Operating and Storage Junction Temperature Range	-55 to +150	°C

*These ratings are limiting values above which the serviceability of any semiconductor device may be impaired.

NOTES:

1) These ratings are based on a maximum junction temperature of 150 degrees C.

2) These are steady state limits. The factory should be consulted on applications involving pulsed or low duty cycle operations.

Thermal Characteristics

T_A = 25°C unless otherwise noted

Symbol	Characteristic	Max	Units
		BC547 / A / B / C	
P_D	Total Device Dissipation Derate above 25°C	625 5.0	mW mW/°C
$R_{\theta JC}$	Thermal Resistance, Junction to Case	83.3	°C/W
$R_{\theta JA}$	Thermal Resistance, Junction to Ambient	200	°C/W

NPN General Purpose Amplifier
(continued)

Electrical Characteristics
TA = 25°C unless otherwise noted

Symbol	Parameter	Test Conditions		Min	Max	Units
OFF CHARACTERISTICS						
$V_{(BR)CEO}$	Collector-Emitter Breakdown Voltage	I_C = 1.0 mA, I_B = 0		45		V
$V_{(BR)CBO}$	Collector-Base Breakdown Voltage	I_C = 10 µA, I_E = 0		50		V
$V_{(BR)CBO}$	Collector-Base Breakdown Voltage	I_C = 10 µA, I_E = 0		30		V
$V_{(BR)EBO}$	Emitter-Base Breakdown Voltage	I_E = 10 µA, I_C = 0		6.0		V
I_{CBO}	Collector Cutoff Current	V_{CB} = 30 V, I_E = 0			15	nA
		V_{CB} = 30 V, I_E = 0, T_A = +150 °C			5.0	µA
ON CHARACTERISTICS						
h_{FE}	DC Current Gain	V_{CE} = 5.0 V, I_C = 2.0 mA	547	110	800	
			547A	110	220	
			547B	200	450	
			547C	420	800	
$V_{CE(sat)}$	Collector-Emitter Saturation Voltage	I_C = 10 mA, I_B = 0.5 mA			0.25	V
		I_C = 100 mA, I_B = 5.0 mA			0.60	V
$V_{BE(on)}$	Base-Emitter On Voltage	V_{CE} = 5.0 V, I_C = 2.0 mA		0.58	0.70	V
		V_{CE} = 5.0 V, I_C = 10 mA			0.77	V
SMALL SIGNAL CHARACTERISTICS						
h_{fe}	Small-Signal Current Gain	I_C = 2.0 mA, V_{CE} = 5.0 V, f = 1.0 kHz		125	900	
NF	Noise Figure	V_{CE} = 5.0 V, I_C = 200 µA, R_S = 2.0 kΩ, f = 1.0 kHz, B_W = 200 Hz			10	dB

Bilag 15

- Standard Dioder

Bilag 16

- Virksomheds opgave

Udvidet kasserapport

								Dato 31/12-2002		Nr. 1
		12410 Kassekonto		15210 Kassekredit		12430 Giro		Modpostering		
Bilag nr	Tekst	Indbetaling	Udbetaling	Indsat	Hævet	Indsat	Hævet	Kt.nr.	Debet	Kredit
630	Kontant varesalg	22250						1100		17800
	Salgsmoms							15272		4450
631	Regning fra lokale avisen				5750			3100	4600	
	Købmoms							15271	1150	
632	Regning fra bogshoppen		550					3900	440	
	Købmoms							15271	110	
633	A-Skat						10328	15250	9180	
	AM-bidrag							15240	1020	
	SP-bidrag							15245	128	
634	Told & Skat						44400	15273	44400	
635	Faktura fra Toyota				150250			11230	120200	
	Købmoms							15271	30050	
636	Privat forbrug		2500					13120	2500	
637	S.Sørensen Konto 122192	15500						122192		15500
638	Løn til bud		424					4100	800	
	AM-bidrag							15240		64
	SP-bidrag							15245		8
	A-Skat							15250		304
639	Indsat på kassekredit		34716	34716						
640	Overført fra kassekredit til girokonto				30000	30000				
	Dagensbevægelser	37750	38190	34716	186000	30000	54728		214578	38126
	Beholdning morgen	2450		420800		58500		12410	37750	38190
	Beholdning aften		2000		269516		33772	15210	34716	186000
	Balance	40200	40190	455516	455516	88500	88500	12430	30000	54728
	Kassedifference		10					3400	10	
	Balance	40200	40200					12410		10
								Balance	317054	317054

Del opgave 2 "bogføring på T-skitser"

H. Hansen 122118	
Debet	Kredit
12250	

15272 Salgsmoms	
Debet	Kredit
	2450
150	
600	

1100 Varesalg	
Debet	Kredit
	9800
600	

12420 Bank	
Debet	Kredit
	1000
	1250

3600 Fragtomkostning	
Debet	Kredit
800	

15721 Købsmoms	
Debet	Kredit
200	
2000	
250	
900	
1050	

B. Bach 122116	
Debet	Kredit
	750

S.Sørensen 152314	
Debet	Kredit
	500

7100 Renteomkost.	
Debet	Kredit
500	

P. Pedersen 152814	
Debet	Kredit
	10000

12110 Varelager	
Debet	Kredit
8000	
1000	

Toyota 152812	
Debet	Kredit
	4500

3500 Bilers drift.omk.	
Debet	Kredit
3600	

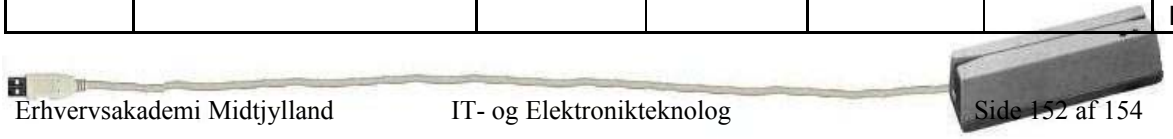
K.Knudsen 122187	
Debet	Kredit
	3000

3300 Tab på tilgode.	
Debet	Kredit
2400	

Tele Danmark 152819	
Debet	Kredit
	5250

3900 Øvrige omk.	
Debet	Kredit
4200	

Konto nr.	Kontonavn	Saldobalance pr. 31/12 2002		Efterposteringer		Resultatopgørelse		Balance	
		Debet	Kredit	Debet	Kredit	Omkostninger	Indtægter	Aktiver	Passiver
1100	Varesalg		kr 6.662,00				kr 6.662,00		
2100	Vareforbrug	kr 2.998,00		kr 162,00		kr 3.160,00			
3100	Salgs. Frem omk.	kr 436,00				kr 436,00			
3200	lokale omk	kr 520,00		kr 45,00	kr 89,00	kr 476,00			
3300	Tab på tilgodeh.	kr 28,00		kr 48,00		kr 76,00			
3900	Øvrige omk	kr 436,00				kr 436,00			
4100	Lønafregning	kr 1.662,00				kr 1.662,00			
4200	ATP bidrag	kr 12,00				kr 12,00			
5300	Afskriv. På Inv	kr -		kr 400,00		kr 400,00			
7100	Rente omk	kr 268,00		kr 50,00		kr 318,00			
7200	Kontant rabat til kunder	kr 144,00				kr 144,00			
11240	inventar	kr 3.102,00						kr 3.102,00	
11241	Akkumulerede afsk. På inv.		kr 1.562,00		kr 400,00				kr 1.962,00
12110	Varelager	kr 2.362,00			kr 162,00			kr 2.200,00	
12210	Vare debitorer	kr 2.522,00			kr 60,00			kr 2.462,00	
12230	Periode afg. Aktiv.			kr 89,00				kr 89,00	
12410	kasse	kr 324,00						kr 324,00	
13110	kapital konto/egenkapital		kr 2.164,00	kr 574,00					kr 1.590,00
13120	Privat forbrug	kr 574,00			kr 574,00			kr -	kr -
15210	kassekredit		kr 1.832,00		kr 50,00				kr 1.882,00
15230	varekreditor		kr 2.498,00						kr 2.498,00
15240	skyldigt AM-bidrag		kr 18,00						kr 18,00
15245	Skyldigt SP-bidrag		kr 2,00						kr 2,00
15250	Skydig A-skat		kr 42,00						kr 42,00
15260	Skyldigt ATP-bidrag		kr 4,00						kr 4,00
15271	Købsmoms	kr 228,00			kr 228,00				
15272	Salgsmoms		kr 394,00	12/382					
15273	Momsafregning			kr 228,00	kr 382,00				kr 154,00
15281	Andre kreditorer		kr 438,00						kr 438,00
15282	Periode afg pas.				kr 45,00				kr 45,00
						kr 7.120,00	kr 6.662,00	kr 8.177,00	kr 8.635,00
					Underskud		kr 458,00	kr 458,00	
						kr 7.120,00	**kr 7.120,00**	**kr 8.635,00**	**kr 8.635,00**



Note	Resultatopgørelsen				Bilag		
	Netto omsætning		kr	6.662,00	**Note 1: Andre eksterne omk**		
	- Vareforbrug		kr	3.160,00	salgsfremmende omk.	kr	436,00
	= Bruttofortjeneste		kr	3.502,00	Lokale omkostninger	kr	476,00
1	- Andre eksterne omk.	kr 1.424,00			Tab på tilgodehavende	kr	76,00
2	- Personale omkostninger	kr 1.674,00	kr	3.098,00	Øvrige omkostninger	kr	436,00
	= Indtjeningsbidrag		kr	404,00	I alt	kr	1.424,00
3	- Afskrivninger		kr	400,00	**Note 2: Personale omkost.**		
	= Resultat før renter		kr	4,00	Lønafregning	kr	1.662,00
	+ Kapital indtægt	kr -			ATP-bidrag	kr	12,00
4	- Kapital omk.	kr 462,00	kr	462,00	I alt	kr	1.674,00
	= Årets resultat			-458,00	**Note 3: Biler & inventar**		
					Biler	kr	-
					inventar	kr	400,00
					I alt	kr	400,00
					Note 4: Kapital omk.		
					Rente omk	kr	318,00
					kontant rabat kunde	kr	144,00
					I alt	kr	462,00

Note	Balance pr 31/12 2002			Bilag		
	AKTIVER:			**Note 5: Anlægsaktiver**		
	Anlægsaktiver:				Biler	Inventar
5	Biler	kr	-	Anskaffelsesværdi	kr -	kr 3.102,00
5	Inventar	kr	1.140,00	- Akkumulerede afskrivning	kr -	kr 1.962,00
	I alt	kr	1.140,00	I alt	kr -	kr 1.140,00
	Omsætningsaktiver:			**Note 6: Tilgodehavende**		
	Varelager	kr	2.200,00	Varedebitor		kr 2.462,00
6	Tilgodehavende	kr	2.551,00	Periodeafgrænsning (Aktiver)		kr 89,00
7	Likvidbeholdning	kr	324,00	I alt		kr 2.551,00
	I alt	kr	5.075,00	**Note 7: Likvidbeholdning**		
	Total	kr	6.215,00	Kasse		kr 324,00
				Bank		kr -
	PASSIVER:			Giro		kr -
	Egenkapital:			I alt		kr 324,00
	Saldo pr. 1/1 2002	kr	2.164,00	**Note 8: Gæld til det offentlige**		
	- Underskud	kr	458,00	Skyldig AM		kr 18,00
	- Privatforbrug	kr	574,00	Skyldig SP		kr 2,00
	= Saldo pr 31/12 2002	kr	1.132,00	Skyldig ATP		kr 4,00
	Gæld:			Skyldig A-Skat		kr 42,00
	Kassekredit	kr	1.882,00	Momsafregning		kr 154,00
	Varekreditor	kr	2.498,00	i alt		kr 220,00
8	Gæld offentlig	kr	220,00			
	Periode afgrænsning (Passiver)	kr	45,00			
	Andre kreditorer	kr	438,00			
	I alt	kr	5.083,00			
	Total	kr	6.215,00