AALBORG UNIVERSITET

Institut for Elektroniske Systemer

Titel: Trådløs kommunikation med PIC MCU

Tema: Microprocessorsystemer

Projektperiode: 1. februar - 31. maj 1001

Storgruppe: E4 2001

Gruppe: 413

Gruppemedlemmer:

Jesper Holst

Eakob Kirkegaard

Morten Tofte Koch

Peder Andersen Lund

Morten Mikkelsen

Esben Juul Sørensen

Vejleder:
Flemming Munk

Oplagstal: 9

Rapport sideantal: 70

Appendiks sideantal: 35

Total sideantal: 105

Afsluttet: 28. maj 2001

Synopsis:

Dette projekt omhandler analyse og syntese af to PIC microcontroller systemer, som kommunikerer bidirektionalt over et radiolink. Det konstruerede system er et fjernbetjeningssystem mellem bruger og bil. Der tages udgangspunkt i nogle af de fordele brugeren kan opnå med bidirektional kommunikation, hvorefter der til projektet er opstillet en afgrænset løsning, som realiseres i praksis.
Fjernbetjeningen er konstrueret med et brugerinterface, der giver mulighed for at indhente data omkring bilen. Herunder informationer om temperatur, benzinstand og status for låsen.
Rapporten beskriver opbygningen af et trådløst kommunikationsystem med tilhørende protokol, en beskrivelse af de to PIC systemer, og de enheder der er tilkoblet dem. Ydermere indeholder rapporten en gennemgang af Seiko L1614 karakterdisplay, Dallas DS1621 temperaturmåler, I^2C protokollen, A/D konvertering baseret på kapacitorstrukturen samt fejlkontrolkodning.

AALBORG UNIVERSITY

Institute of Electronic Systems

Title: Wireless communication using PIC MCU

Theme: Microprocessor systems

Time period: 1st of February - 31st of May 2001

Term: E4 2001

Group: 413

Members:

Jesper Holst

Jakob Kirkegaard

Morten Tofte Koch

Peder Andersen Lund

Morten Mikkelsen

Esben Juul Sørensen

Instructor:
Flemming Munk

Number printed: 9

Number of pages: 70

Appendix pages: 35

Total number of pages: 105

Ended: 28th of May 2001

Abstract:

The aim of this project is to perform an analysis and synthesis of a microcontroller system based on the PIC architecture. The microcontroller systems communicate bidirectionally through a radiolink, and is thereby implementing a master/slave remote control system between the user and the car. The starting point of the project is to examine advantages of bidirectional communication in preference to normal one way communication.
The remotecontrol is equipped with a user interface, which implements the possibilty of collecting data gathered in the car. Among these is informations concerning the temperature, state of the fuel tank and state of the locks.
This report contains the construction of the wireless communication system with an integrated protocol, a description of the PIC systems and the units attached to the systems. Furthermore the report contains an examination of the Seiko L1614 character display, Dallas DS1621 thermometer probe, the I^2C protocol, the A/D converter based on the capacitor structure, together with error correcting codes.

Indhold

Forord

1

Dette projekt er udarbejdet af gruppe 413 ved Institut for Elektroniske Systemer på Aalborg Universitet. Projektet er udarbejdet i P4-perioden, der strækker sig fra den 1. februar til den 31. maj 2001. Det overordnede formål med P4-projektet er at få den tilførte tekniske viden fra PE-kurserne indført i den problemorienterede og projektorganiserede indlæringsform gennemført i grupper.

Målgruppen for dette projekt er studerende med et teknisk niveau svarende til gruppens eget.

I rapporten er alle litteraturhenvisninger udført i henhold til Harvardmetoden, dvs. forfatter og årstal står i kantet parentes - eks. [Sedra & Smith, 1998]. Placering af henvisningen afgør, hvad der specifikt henvises til. Står kildehenvisningen før et punktum, refereres der til den foregående sætning - står henvisningen efter et punktum, er det hele det foregående afsnit. Figurer og ligninger er nummereret efter kapitel og et fortløbende nummer. Eksempelvis kan figur 1.10 findes i kapitel 1, som den 10. figur. Desuden benyttes punktum, som decimalseparator, da dette benyttes i de fleste simulerings- og matematikprogrammer. Vektorer noteres med fed, som $\mathbf{A}$, mens matricer noteres med fed og overstreget, som $\overline{\mathbf{A}}$. Alle anvendte datablade samt rapport og kildekode er vedlagt på cd-rom.

Aalborg d. 28. maj 2001

Jesper Holst	Jakob Kirkegaard	Morten Tofte Koch
Peder Andersen Lund	Morten Mikkelsen	Esben Juul Sørensen

Indledning 2

Formålet med dette kapitel er kort at introducere problemstillingen, som behandles i projektet, for derefter at analysere det valgte problem. Dette gøres med henblik på at opstille en endelig problemformulering for projektet og for senere at kunne foretage en præcis kravsspecifikation for systemet, som ønskes opbygget.

De første maskiner, der blev styret og kontrolleret vha. fjernbetjeninger, blev primært benyttet i forbindelse med militære formål. Allerede under 1. verdenskrig benyttede den tyske flåde sig af radiostyrede motorbåde, mens begge parter under 2. verdenskrig gjorde brug af radiostyrede bomber samt andre fjernstyrede våben. Efter 2. verdenskrig begyndte amerikanske videnskabsmænd at eksperimentere med mere fredelige anvendelsesmuligheder af fjernbetjeningen. Sidst i 1940'erne blev fjernbetjente garagedøre opfundet, mens der i begyndelsen af 1950'erne anvendtes TV fjernbetjeninger. [Julie, 1994]

Siden da er der fundet mange andre anvendelser af fjernbetjeninger inden for adskillige områder i hverdagen - eksempelvis i forbindelse med hi-fi udstyr, overvågning, klimanlæg, biler samt mange andre områder, hvor det er praktisk at kunne overvåge og kontrollere over kort eller længere afstand. Fælles for de nævnte fjernbetjeningssystemer er, at der for det meste er tale om envejskommunikationen. I langt de fleste tilfælde er dette tilstrækkeligt, idet systemerne, som kontrolleres og styres, ikke har nogen væsentlig information at sende tilbage til brugeren.

Dette projekt sigter imod at undersøge hvilke muligheder, der ligger i at benytte tovejskommunikation med henblik på udveksling af relevante informationer imellem mastersystemet (fjernbetjeningen) og slavesystemet (systemet som fjernstyres). Informationerne, der overføres fra slavesystemet til master systemet, kan eksempelvis være data opsamlet i forbindelse med målinger på fysiske processer eller informationer om hændelser ved slave systemet.

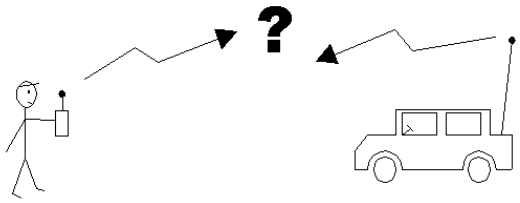
I forbindelse med undersøgelsen af mulighederne i tovejskommunikation, har gruppen som udgangspunkt valgt at anvende et fjernbetjeningssystem til en bil med henblik på at afdække hvilke interessante informationer, der kan udveksles, og hvordan systemerne kan vekselvirke med henblik på større komfort, sikkerhed etc. for brugeren af bilen. Applikationsområdet for systemerne er således en bil.

Dette fører frem til det af gruppen valgte initierende problem:

Hvorledes konstrueres to microprocessorsystemer, som kan kommunikere bidirektionalt, sikkert og med lav sandsynlighed for fejl via et radiolink, og kan denne øgede interaktion mellem systemerne udnyttes i forbindelse med automotive fjernbetjeninger ?

2.1 Problemanalyse

Formålet med dette afsnit er at afdække hvilke aspekter ved det valgte fjernbetjeningssystem, som med fordel kan benytte sig af bidirektional kommunikation. Dette sker indledningsvis igennem en analyse af mulighederne for dataopsamling etc. i bilen, samt for at introducere eventuelt nye egenskaber i fjernbetjeningen. Dette gøres med henblik på at undersøge mulighederne, som ligger i den øgede interaktion mellem master og slave systemet for endelig at kunne opstille en problemformulering samt en kravsspecifikation for projektet.



Figur 2.1: Fjernbetjeningssystem.

2.1.1 Bilen

I forbindelse med diskussioner i gruppen er følgende ting blevet nævnt, som mulige udvidelser af det system, der i øjeblikket kendes fra biler - dvs. en fjernbetjening til op- og aflåsning af bilen.

Overvågning. Forskellige sensorer kan indbygges i bilen til overvågning af diverse processer, i forbindelse med alarmering, i tilfælde af indbrud, eller for give brugeren mulighed for at aflæse status af forskellige parametre i bilen. Følgende sensorer er overvejet anvendt:

- Closed-loop sensorer, til monitorering af eks. døre som åbnes, eller andre ting der afbrydes.
- Lydsensorer til detektion af glas, der bliver knust eller lignende.
- Bevægelsessensorer til detektion af personer i og omkring bilen.

- Kameraer, som kan sende billeder tilbage af personer i bilen.

Komfort og sikkerhed. Identifikation vha. fjernbetjeningen giver mulighed for personlig indstilling af bilens elektroniske udstyr. Eksempelvis indstilling af førersæde, selehøjde, instrumentlayout, aircondition, bakspejl, sidespejle, rat, motorstyring og støddæmpning. Afhængigt af hvilken person, der identificerer sig med fjernbetjeningen, vil aktuatorer automatisk indstille det pågældende udstyr.

Ulykker. Ved ulykker kan bilen blive i stand til at tilkalde ambulance samt transmittere data vedrørende antal personer i bilen. Dette kan gøres med henblik på en mere effektiv redningsindsats.

2.1.2 Fjernbetjeningen

I forbindelse med fjernbetjeningen er følgende ting blevet overvejet:

Overvågning. Fjernbetjeningen kan give mulighed for at overvåge bilen og modtage forskellige parametre fra denne. Dette kan eksempelvis være status for låsen, temperaturen i bilen samt positionen.

Kontrol. Mulighed for op- og aflåsning af bilen, samt styring af alarmeringstransducer i bilen.

Interface. Via et interface bestående af en eller flere knapper samt et display, kan der opnås høj brugervenlighed af fjernbetjeningens tilgængelige funktioner. Der kan opbygges et menusystem på displayet, som kan styres af knapper. Herigennem bliver det muligt for brugeren at overvåge og kontrollere bilen.

2.2 Problemformulering

På baggrund af ovenstående diskussion om hvilke krav, som kan være relevante at stille til fjernbetjeningssystemet, samt de overordnede krav, der stilles i forbindelse med projekttemaet, har dette projekt til formål at opbygge et microprocessorsystem, hvor ønskes følgende spørgsmål besvaret:

1. Hvorledes kan der etableres en bidirektional kommunikation mellem to microprocessorsystemer i et master/slave forhold?

2. Hvordan kan der udvikles en passende kommunikationsprotokol?

3. Hvordan kan der i kommunikationen mellem systemerne udvikles en metode, som i størst mulig omfang sikrer, at informationerne, der udveksles, kan transmitteres med lav sandsynlighed for fejl samt detektere og evt. rette et passende antal fejl?

4. Hvordan kan microprocessorsystemet i fjernbetjeningen fungere som et interface mellem bruger og det andet microprocessorsystem?

5. Hvilke opgaver kan microprocessorsystemet i bilen med fordel foretage målinger på?

6. Hvorledes skal målingerne opsamles og behandles af microprocessorsystemet?

2.3 Kravsspecifikation

Ud fra hovedtrækkene for de systemer som ønskes konstrueret i forbindelse med dette projekt, vil dette afsnit beskæftige sig med specifikke krav til systemerne. Dette vil sige elektriske, kommunikations- samt interfacemæssige krav.

2.3.1 Microprocessorsystemer

For microprocessorsystemerne gælder, at de skal kunne foretage målinger på omgivelserne, samt være i stand til at foretage målinger på forskellige processer simultant. Dette medfører følgende krav til systemerne:

Interface. Systemerne skal give mulighed for et parallelt I/O interface, således det er muligt at kommunikere med eksterne enheder som display, sensorer eller andre microprocessorsystemer.

Interrupthåndtering. For at udnytte processorens regnekraft mest muligt og optimere hastigheden af programafviklingen, skal systemet kunne håndtere interrupts. Til tidskritiske systemer som radiokommunikation skal der være mulighed for et timerbaseret interrupt, så dataoverførsel mellem de to systemer kan synkroniseres.

ADC. Systemet, der implementeres i bilen, skal give mulighed for A/D konvertering, såfremt der påkobles sensorer med analoge output.

Effektforbrug. Især for fjernbetjeningssystemet ønskes et begrænset effektforbrug og så lav en spænding som mulig, så fjernbetjeningen kan drives af et enkelt batteri med længst mulig levetid.

2.3.2 Kommunikation

Kommunikationen mellem bil og fjernbetjening skal rent fysisk foregå via et radiolink, som benytter det licensfrie frekvensområde. Dvs. bil og fjernbetjening skal udstyres med transceiver moduler, som er i stand til at kommunikere half duplex i et master/slave forhold. Kommunikationen skal foregå serielt efter en protokol. Dette afsnit beskriver generelle krav til denne protokol.

Fejlkorrektion. Et fejlkorrektionssystem ønskes implementeret.

Datapakker. Ved hver transmission ønskes at sende et antal bytes, som giver mulighed for at sende en databyte samt kommando- og fejlkorrektionsbits.

Hastighed. Under hensyntagen til hardwaremæssige begrænsninger ønskes så hurtig en kommunikation som muligt. Der tages udgangspunkt i en overførselshastighed på ca. 9600 [baud], der anses at være tilstrækkelig i forhold til den mængde data, der skal overføres. Effektmæssigt er det ligeledes hensigtsmæssigt at sende data så hurtigt som muligt, idet radiomodulerne således er aktive i kortere tid.

Effektforbrug. Imellem transmissioner skal det være muligt at sætte modulerne på standby, hvorefter de vækkes med en bestemt dutycycle i et fastlagt tidsinterval for at detektere indkommende radiosignaler. Dette gøres ligeledes for at spare effekt.

Der skal kunne benyttes samme forsyningsspænding for alt periferiudstyr, som for microprocessorsystemerne.

Rækkevidde. Rækkevidden defineres til at være den afstand i åbent terræn, over hvilken delsystemerne kan kommunikere, uden hver datapakke skal retransmitteres.
En rækkevidde mellem fjernbetjening og bil i åbent terræn på 50 [m] anses som værende tilstrækkelig.

2.3.3 Interface

Med interface menes grænsefladen mellem brugeren og microprocessorsystemet. Kravene til interfacet har til formål at skabe forøget brugervenlighed.

Display. For at øge overskueligheden af fjernbetjeningens muligheder skal der benyttes et karakterdisplay med et let overskueligt menusystem til visning af forskellige parametre. På denne måde kan brugeren følge med i fjernsystemets tilstand.

For at optimere microprocessorsystemets I/O kapacitet skal kommunikationen mellem displayet og microprocessorsystemet foregå serielt, eventuelt gennem et skifteregister. Displayet skal kunne operere med samme forsyningsspænding som microprocessorsystemerne.

Knapper. På fjernbetjeningen skal der være to knapper, til henholdsvis aflåsning og udlæsning af statusparametre. Følgende parametre skal kunne udlæses: Låsestatus, benzinstand, alarm og temperatur.

2.3.4 Sensorer og aktuatorer

For at opnå den ønskede funktionalitet skal der placeres sensorer og aktuatorer flere steder i bilen. Det kræves, at disse enheder skal være spændingsmæssig kompatible med microprocessorsystemet, så yderligere spændingsomformning ikke er nødvendig.
Der skal ved hjælp af en sensor, være mulighed for at kunne udlæse en temperatur, ligesom en fiktiv benzinstand skal kunne udlæses.

2.4 Problemafgrænsning

På grundlag af kravsspecifikationen, og ud fra et teknisk synspunkt, vælges enhederne, der skal udgøre det samlede system. Systemet vil ikke blive implementeret i en bil, men fremstillet som en prototype, hvor den anvendte teknik danner en basis for et fuldt system.
Der vil derfor blive konstrueret et system, der teknisk set løser den opstillede problemstilling, men hvor visse komponenter kun har til formål at simulere output fra eksterne delsystemer. Valget af komponenter til systemet begrundes i de følgende afsnit.

Microprocessorsystem. Til dette projekt vælges to mid-range PIC MCU[1] fra Microchip. En MCU kombinerer et microprocessorsystem med hukommelse og I/O interface til eksterne enheder i ét integreret kredsløb. PIC teknologien baseret på RISC[2] og Harvard struktur giver samtidig en hurtig og kosteffektiv MCU. [Microchip-Midrange, 1997]

I fjernbetjeningssystemet vælges en PIC16F84A på grund af dens fysiske størrelse og dens lave effektforbrug. MCUen har to porte bestående af 13 I/O ben, interrupthåndtering, samt mulighed for at køre med en clockfrekvens på op til 20 [MHz].

Til delsystemet i bilen vælges en PIC16F877. Denne MCU vælges, idet den har en stor hukommelse, stor I/O kapacitet, ADC, samt understøttelse for seriel kommunikation. Fysisk er den større, men da dette er uden relevans, for systemet placeret i bilen, er valget uproblematisk.

Til begge delsystemer vælges en forsyningsspænding på 5.0 [V], som også kan benyttes til display og radiomodulerne, der kræver en forsyningsspænding mellem 4.5-5.5 [V].

Kommunikation. Til den trådløse kommunikation vælges to BIM-433-F transceivere fra Radiometrix, der kan kommunikere i half duplex. Disse har under optimale forhold en rækkevidde på op til 120 [m] i åbent terræn, ligesom det er muligt at transmittere med op til 40

[1] MCU: Micro Controller Unit
[2] RISC: Reduced Instruction Set Computer

[kbaud]. En sleep funktion og en hurtig opvågning på < 1 [ms] sikrer et minimalt effektforbrug.

Display. Som en del af interfacet vælges et Seiko L1614 display, men alternativt kunne der vælges et mindre display, så det kunne implementeres i en nøglering eller i selve nøglen. Til dette projekts prototype er størrelsen af det valgte display acceptabel.

Sensorer. Sensorerne og aktuatorerne bliver ikke implementeret i en bil. Enkelte sensorer og aktuatorer vælges for at vise princippet, og der tages ikke hensyn til eventuelle pladsbegrænsninger, ledningsnet etc. i bilen. Måling af benzinbeholderens tilstand simuleres med et potentiometer, der er tilsluttet PICens ADC, for at vise princippet bag konvertering af analoge spændinger til digitale. Oplysning om bilens temperatur foretages af en Dallas DS1621 temperatur chip. Denne udmærker sig ved af have indbygget ADC, og kan kommunikere via I^2C bussystemet.

Konstruktion 3

I det følgende afsnit defineres et minimumssystem til en microcontroller, samt den grundlæggende styring af microprocessorsystemet. Opdelingen mellem to microprocessorsystemer vil ligeledes fremgå.

3.1 Minimumssystem

Hardwaremæssigt er det nødvendigt, at flere fysiske rammer er opstillede, for at microprocessoren kan fungere. Nogle af de væsentligste krav er skitseret her:

- Korrekt forsyningsspænding $V_{DD}(5\pm\ 0.5[\mathrm{V}])$.
- Kompatibel oscillator konfiguration og hastighed.
- Korrekt håndtering af masterclear ($\overline{MCLR}$).

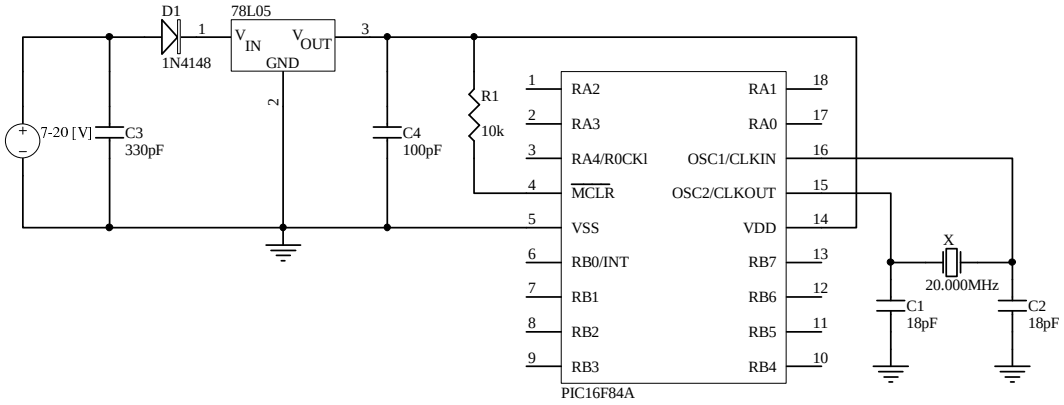
Forsyningsspænding. Forsyningsspændingen er først og fremmest beskyttet mod forkert polarisering, ved hjælp af en standard signaldiode. Ydermere er selve spændingen stabiliseret ved brug af en spændingsregulator. I dette tilfælde er en LM78L05 fra National Semiconductors valgt. Denne komponent er velegnet til småsignalkredsløb med lave effektforbrug. LM78L05 kan spændingsregulere fra 7 til 20 [V] med en kontinuert arbejdsstrøm på 100[mA] (peakstrømme op til 140 [mA]) ned til de påkrævede 5 [V]. Processen foregår internt i en trebenet IC indeholdende knap 20 transistorer, hvilket resulterer i, at spændingen holdes på 5 ± 0.25[V], som tilfredsstiller kravet fra den valgte PIC (5 ± 0.5[V]). ICen skal ifølge databladet udstyres med ind- og udgangskapacitorer på hhv. 330 og 100 [pF]. Dette sker for at dæmpe støj fra reguleringsprocessen. [National-Semiconductors, 2000]

Oscillator. PIC processoren kan arbejde med forskellige oscillatortyper, disse kan opdeles i fire forskellige kategorier:

1. RC : Resistor-Kapacitor kan kun anvendes op til 4 [MHz] clockfrekvens og kan ikke give præcis timing.
2. LP : Low Power kan ikke anvendes for 20 [MHz] PIC.
3. XT : XTal som betegner en keramisk resonator (et krystal) kan ligeledes kun anvendes op til 4 [MHz] clockfrekvens, men giver til gengæld en præcis timing.
4. HS : High Speed oscillator anvender også en keramisk resonator, men konstellationen kan køre op til 20 [MHz] clockfrekvens, mod et større strømforbrug.

Der er imidlertid behov for en hurtig clockfrekvens, da processoren skal kunne udføre relativt komplekse rutiner, uden det må påvirke timingen for eksterne kredsløb i nævneværdig grad. Der vælges derfor at benytte HS-oscillatoren. Den opbygges ved hjælp af et krystal med 20.000 [MHz] resonansfrekvens, samt to kapacitorer på mellem 15 og 33 [pF]. [Microchip-16F84A, 1998]

Masterclear. Masterclear ($\overline{MCLR}$) skal, som det ses af symbolet, negeres for at være aktiv. Når processoren skal udføre instruktioner, skal masterclear med andre ord være høj. Det påregnes ikke at anvende masterclear, når processorsystemet er færdigudviklet, hvorfor $\overline{MCLR}$ hardwires til V_{DD} ved hjælp af en 10 [kΩ] pull up resistor.
Et diagram over det opbyggede minimussystem kan ses på figur 3.1.



Figur 3.1: Elektrisk diagram over det opbyggede minimussystem, her vist med PIC16F84A-20.

3.2 Grundlæggende styring

Til at styre processoren er der valgt at skelne mellem to forskellige procestyper:

- Interruptstyret programafvikling.

- Almindelig programafvikling.

Interruptstyring. Interruptstyring udmærker sig ved, at en proces kan udføres ved at afbryde en igangværende proces. PIC microprocessoren har kun ét interruptniveau, hvilket, hvis der ønskes flere niveauer, betyder at der skal laves software interrupts[1]. Denne metode er besværlig og svær at overskue, hvorfor der er valgt kun at bruge ét interrupt. Dette interrupt reserveres til radiokommunikationen, da denne er særdeles tidskritisk. Det er med andre ord meget vigtigt, at timingen ikke afbrydes, når data skal sendes og modtages.

Almindelig afvikling. Almindelig programafvikling anvendes således til alle andre formål. I stil med processorens egen Ifetch-løkke[2], udvikles en main-løkke, som gennemløber forskellige procedurer og handler derefter. Der fås herved ikke en afvikling af de forskellige procedurer efter behov, men efter tur, hvorfor incitamentet for at vælge en hurtig processor er klart.
Main-løkken, er den eneste del af programmet, som skal styre afviklingen af procedurene i processoren og tillade interrupts. Opbygningen kan kortfattet beskrives som:

- Initialisering.
- Poll-løkke.
- Pause.

Initialiseringen. Initialiseringen gennemkøres, når processoren opstartes eller resettes. Denne del af main-løkken opsætter porte, interrupts og andet PIC relateret hardware. Efterfølgende initialiseres periferihardwaren.

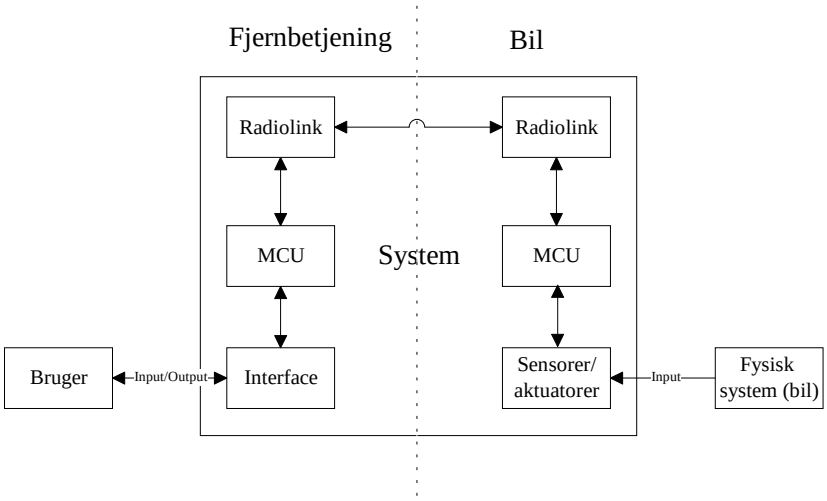
Poll-løkken. Denne løkke består af procedurer, som udføres, når processoren er initialiseret og kører. Løkken koordinerer de forskellige ressourcer, som er tilknyttet processoren, således at der f.eks. ikke både modtages og sendes på samme tid. Efter et gennemløb af poll-løkken udføres en pause.

Pause. En pause afholdes, ved at sætte processoren i busy-waiting, hvilket vil sige at den sættes til at udføre ligegyldige instruktioner, indtil et foruddefineret tidsrum er gået. Processoren kan i denne tid ikke servicere andet end de procedurer som er tildelt interruptstatus. Det kan virke uhensigtsmæssigt, at processoren ikke kontrollerer om en knap er aktiveret så ofte som muligt, men da pausen kun varer ca. 90 [ms], er det uden praktisk betydning for brugeren. Til gengæld kan periferihardware slukkes eller sættes på standby i pausen, så der kan opnås en ikke ubetydelig energibesparelse.

[1] Programmøren skal selv skelne mellem hvilke interrupts, der skal udføre hvad.
[2] Instruction fetch - Intern styring til programafvikling i en CPU.

3.3 Opdelingen mellem to microprocessorsystemer

Systemet kan overordnet inddeles i to delsystemer bestående af en fjernbetjening (master) og alarmen i bilen (slave). Fjernbetjeningen består af et brugerinterface, en MCU og et radiolink, og delsystemet i bilen består af en MCU, et radiolink og nogle sensorer og aktuatorer. Det er således muligt at afgrænse systemet fra omgivelserne samt definere I/O til og fra systemet. Dette er vist på figur 3.2.

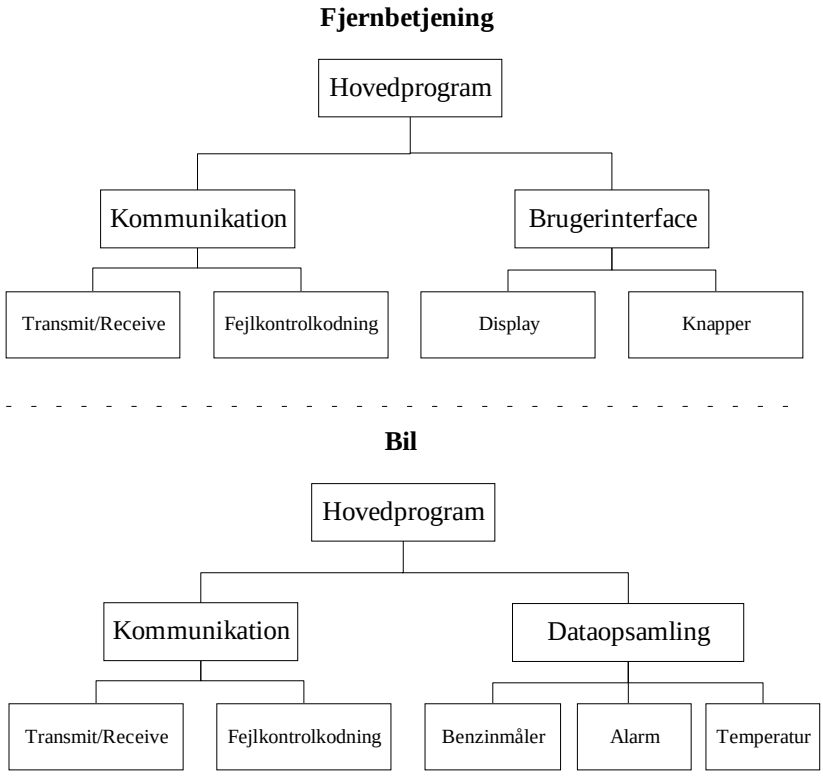


Figur 3.2: Afgrænsning af systemet.

Selve systemet er inddelt i delsystemer, der hver varetager en overordnet funktion i systemet. Delsystemerne repræsenterer således en mængde hardware som specificeret i problemafgrænsningen. Med udgangspunkt i denne inddeling udvikles et eller flere softwaremoduler for hvert delsystem indeholdende drivere til styring af den tilhørende hardware. Derudover kan der udvikles mindre hardwarerelateret softwaremoduler til eksempelvis kodning og dekodning af data.

Programafviklingen styres af hovedprogrammet i hver MCU ved kald af de enkelte softwaremoduler. Modulerne kan derfor opfattes som funktioner, der kaldes, når de ønskes udført. Til hver funktion knytter der sig specifikke ind- og udgangsvariable.

På grundlag af opdelingen af systemet defineres grænseflader i systemet ved inddeling af softwaren i følgende moduler:



Figur 3.3: Inddeling af software i moduler.

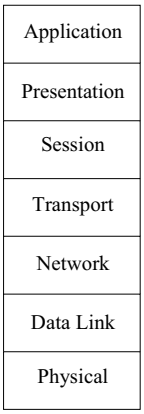
I de følgende afsnit vil konstruktionen samt funktionaliteten af softwaremodulerne beskrives, ligesom ind- og udgangsvariable defineres. Afsnittene tager udgangspunkt i figur 3.3.

Kommunikation 4

I henhold til kravene til den trådløse kommunikation udvikles en protokol. Dette afsnit beskriver protokollen i detaljer, og hvorledes den implementeres med den anvendte hardware.

4.1 Protokol

Open System Interconnection (OSI) grundlagt af International Organization for Standardization har specificeret en referencemodel for protokoludvikling. Modellen tager udgangspunkt i en inddeling af protokoller i 7 lag som på figur 4.1.



Figur 4.1: OSI referencemodel for protokollag

Omfanget af protokollen i dette projekt er forholdsvis begrænset, idet kommunikationen foregår mellem to delsystemer med identiske systemkomponenter og identisk systemarkitektur. Uden at omtale de enkelte lag i OSI reference modellen inddeles protokollen i dette projekt ud fra modellen i 3 lag. De tre lag defineres følgelig:

- Fysisk lag: Beskrivelse af grundlæggende regler for den fysiske transport af de enkelte bits.
- Data lag: Specifikation af datapakker samt flowkontrol herunder sikkerhed og pålidelighed af de overførte data.
- Applikationslag: Specifikation af ind- og udgangsregistre samt kontrolregistre.

Fysisk lag. Datatransmissionen foregår med de valgte BIM-433-F transceivermoduler. På baggrund af databladet for BIM-433-F vælges dette protokollag således til at indeholde følgende krav:

- 5 [V] CMOS interface.
- Half duplex datatransmission.
- Datatransmission med en frekvens på 433.920 [MHz] i det frie licensbånd.

- Manchesterkodning[1] af databits.

Data lag. Omfanget af data, der ønskes sendt, er begrænset. Der vælges derfor en fast pakkestørrelse på 2 bytes. Én byte reserveres primært til overførsel af data. Såfremt der ikke sendes data, kan denne byte indeholde kommandobits. Den anden byte skal indeholde 3 kommandobits og 5 fejlkorrektionsbits. Der er således mulighed for at sende 2^3 forskellige kommandoer samtidig med overførsel af en databyte.

Flow kontrollen har til formål at sikre en korrekt afvikling af datatransmission. Dette kræver en præcis timing af transmissionen. Derfor sendes i starten af hver datapakke en preamble, der etablerer og stabiliserer forbindelsen efterfulgt af et antal startbits til synkronisering af samplingsfrekvensen i transceivermodulerne. Derefter følger datapakken.

Varigheden af preamblen er bestemt af tidsrummet mellem aktivering af radiomodulerne, og tiden der går, fra en bærebølge detekteres, til forbindelsen er stabil. Ud fra hensyn til hastigheden af kommunikationen vælges en frekvens f_{RDO} på

$$f_{RDO} = \frac{1}{90[ms]} = 11[Hz] \tag{4.1}$$

Da radiomodulet maksimalt kræver 4 [ms] til at vågne og stabilisere sig, kræves en varighed for preamblen >94 [ms], der således er tilstrækkelig for at skabe en bærebølge. Databladet for BIM-433-F specificerer et bitmønster for preamblen givet ved 10101010.....

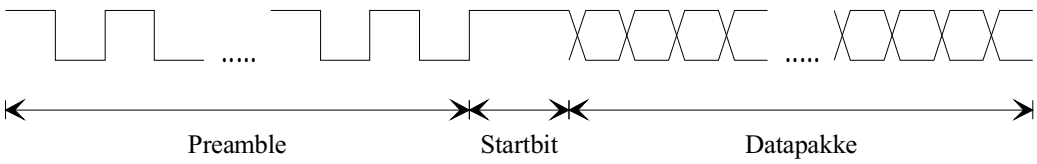
Når transmissionen er færdig, skal datapakken behandles hurtigst muligt. For dette system har det imidlertid ikke nogen indflydelse på sikkerheden, at de sendte data ikke behandles i realtid.

Opsummeret er følgende punkter indeholdt i protokollen i data link laget:

- Fast størrelse af datapakker på 2 bytes med 8 databits, 3 kommandobits og 5 fejlkorrektionsbits.
- Transmission af preamble på >94 [ms] forud for hver transmission.
- 2 høje startbits umiddelbart før transmission af datapakke.
- Fejlkontrolkodning.
- Soft deadline for databehandling, hvor datapakken behandles hurtigst muligt uden krav til behandling i realtid.

[1] For at opnå symmetri mellem antallet af logisk høje og logisk lave bits der transmitteres kan data manchesterkodes. Således vil 0 kodes til 01 og 1 kodes til 10

Dataflowet under transmission ses på figur 4.2:



Figur 4.2: Dataflow.

Applikationslag. Når data skal transmitteres via radiolinket, gemmes de to databytes, der ønskes sendt, i to registre TX0 og TX1. Transmissionen kan efterfølgende initialiseres ved et procedurekald i hovedprogrammet. Modtagelsen af data sker automatisk, og datapakken lægges i to registre RX0 og RX1.

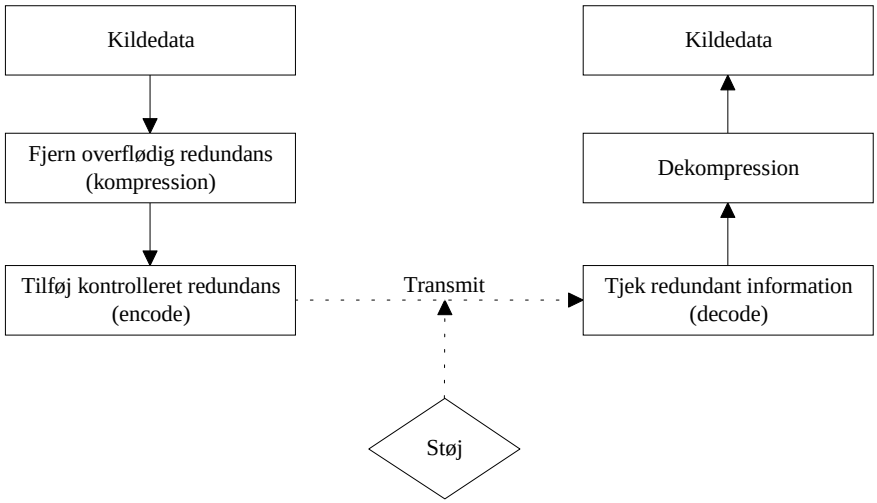
Derudover findes to kontrolregistre, hvor det er muligt for delsystemerne at aflæse status for afsendelse og modtagelse under og efter transmission. Kontrolregistrene omtales nærmere i afsnittene 4.3 og 4.4.

4.2 Fejlkontrolkodning

Protokollen specificerer, at der skal benyttes fejlkontrolkodning. Formålet med fejlkontrolkodning er at mindske sandsynligheden for, at der sker fejl, når data sendes over en given kanal, der pga. ydre påvirkninger introducerer støj af forskellig karakter.

Indtil 1945 var overbevisningen, at støj på kommunikationslinjer var en integreret og uundgåelig forhindring, som under alle forhold ville påvirke de data, som blev sendt over kanalen. Dette blev dog modbevist af Claude Shannon, som viste, at der ved en tilstrækkelig sænkning af transmissionshastigheden, kan opnås en fejlfri transmission i en vilkårlig støjfyldt kanal. For digitale signaler vil en sænkning af hastigheden være ækvivalent med en forlængelse af de koder som sendes. Blot koderne gøres længere, kan der i teorien opnås en fejlfri transmission. [Ebert, 2000]

I praksis gøres koderne længere ved at tilføje en vis mængde kontrolleret redundans inden transmissionen, som gør modtageren i stand til at vurdere gyldigheden af de modtagne data. Princippet i dette er vist på figur 4.3.



Figur 4.3: Manipulation af information

4.2.1 Kanalmodel

I det følgende beskrives den kanal, data ønskes transmitteret over. Herunder gøres en række antagelser, udfra hvilke kodningsstrategien kan vælges.

Det antages, at der er tale om en **random error** kanal, hvilket vil sige, at der for ethvert sæt af symboler a og b, eksisterer en fast sandsynlighed $p_{a,b}$ for, at når a transmitteres, så modtages b. Den vigtigste egenskab ved denne kanalbeskrivelse er, at $p_{a,b}$ ikke afhænger af, om det foregående symbol er transmitteret korrekt eller andre tidligere hændelser i systemet.

Derudover antages kanalen at være **symmetrisk**, hvilket vil sige, at $p_{a,b}$ (sandsynligheden for at modtage b, når a transmitteres) er ens for alle valg af a og b, hvor $a \neq b$.

4.2.2 Typer af koder

Inden for kodningsteorien findes to hovedtyper af koder, idet der tales om:

Blokkoder. Ved denne kodningsform tilføjes hver blok af informationsbit en række paritetsbit, som kun afhænger af de pågældende infomationsbit. Kendte koder af denne type er Hamming og Golay koder.

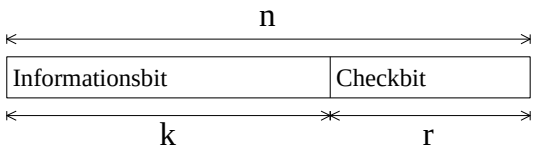
Trækoder. Ved denne kodningsform afhænger paritetsbitene ikke udelukkende af den akuelle blok af informationsbit, men også af tidligere kodeblokke. Foldningskoder er en specialtype af trækoder, og disse benyttes bla. i forbindelse med fejlkontrolkodning på cder (Reed-Solomon kodning).

Ud fra kanalmodellen, som blev opstillet i foregående afsnit (4.2.1), samt matematisk kompleksitet, vælges blokkoderne som kodningsform.

For begge kodetyper gælder, at de både er i stand til at detektere og rette fejl. Hovedsageligt kan koderne anvendes i forbindelse med fejldetektion med henblik på ARQ[2] eller fejlkorrektion i forbindelse med FEC[3]. [Ebert, 2000]
I forbindelse med konstruktion af fejlkontrolkodning til dette projekt ønskes en kombination af ARQ og FEC.

4.2.3 Lineær blokkode

For blokkoder gælder, at data transmitteres i individuelle blokke på n bit, hvor hver blok består af *k* informationsbit og *r* checkbit, som er linearkombinationer af de pågældende informationsbit. Hvis disse bit er ordnet som vist på figur 4.4, hvor informationsbit og checkbit er adskilt i blokke, kaldes koden systematisk.



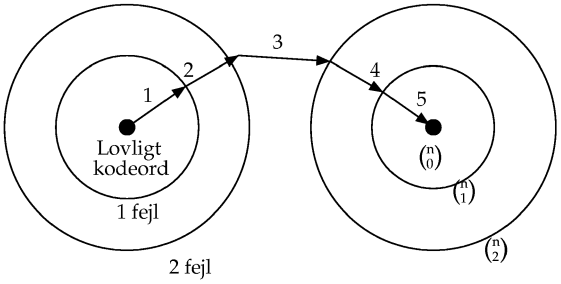
Figur 4.4: Systematisk format af kode

Hamming vægt og afstand. Et kodeords Hamming vægt er antallet af 1-taller i kodeordet, mens Hamming afstand mellem to datavektorer er antallet af forskellige bit. Det kan vises, at Hamming afstanden mellem to datavektorer er lig med Hamming vægten af deres modulo-2 sum (xor). Ud fra Hamming afstanden defineres en kodes minimums afstand, idet denne er den mindste Hamming vægt af et lovligt kodeord - bortset fra nulvektoren. Denne afstand betegnes d_{min} og afgør kodens egenskaber i forbindelse med fejl detektion og korrektion. [Pretzel, 1992]

Fejl detektion og korrektion. En blokkode med d_{min} kan detektere $d_{min} - 1$ fejl og rette $\frac{d_{min}-1}{2}$ fejl - dog ikke samtidigt. Dette er eksemplificeret ved figur 4.5, hvor en kode med

[2] Automatic Repeat reQuest.
[3] Forward Error Control.

$d_{min} = 5$ er illustreret. Det venstre centrum afbilder den korrekte datavektor, som blev transmitteret. Alt efter hvor mange fejl, der sker i forbindelse med transmissionen, bevæger datavektoren sig længere over mod det højre centrum. Ved 1 eller 2 fejl er datavektoren stadig tættere på det venstre centrum og fejlen kan rettes. Ved 3 fejl er datavektorens afstand mindre til det andet centrum, hvorfor fejlen ikke kan rettes. 4 fejl opfattes som var der sket en fejl i forhold til det højre centrum. [Pretzel, 1992]



Figur 4.5: Princippet i fejl-detektion og -korrektion illustreret ved en kode med $d_{min} = 5$. Cirklernes centre betegner lovlige kodevektorer.

Kodning. Ud fra antallet af informationsbit og checkbit defineres koden, der kaldes en (n,k) blokkode med effektivitet $\frac{k}{n}$. Udfra dette kan det således ses, at for en (n,k) blokkode eksisterer der:

- 2^n mulige datavektorer
- 2^k lovlige datavektorer

Hvis alle lovlige datavektorer er linearkombinationer af hinanden, samt at nulvektoren er en lovlig kodevektor, kaldes blokkoden lineær. Det kan således ses, at en **kodning** af en lineær (n,k) blokkode er en lineær afbildning af $\mathbf{B}^k$ over i $\mathbf{B}^n$. Kodningsfunktionen E defineres efterfølgende som en lineær transformation i form af matrix multiplikation:

$$\mathbf{C} = E(\mathbf{D}) = \overline{\mathbf{G}} \cdot \mathbf{D} \tag{4.2}$$

hvor **C** er den kodede og **D** den ukodede datavektor. Begge er søjlevektorer med dimensionerne n henholdsvis k. $\overline{\mathbf{G}}$ er en $n \times k$ matrix. Hvis denne generator matrix består af en $k \times k$ enhedsmatrix og en $r \times k$ paritetsmatrix, genererer matricen systematisk kode.

$$\overline{\mathbf{G}} = \begin{bmatrix} I_{k \times k} \\ P_{r \times k} \end{bmatrix}_{n \times k} \tag{4.3}$$

Der kan findes flere forskellige $n \times k$ matricer til den samme kode, men ikke alle $n \times k$ matricer er generator matricer for en kode. For at en $n \times k$ matrix kan være generator for en

lineær kode, skal søjlerne i $\overline{\mathbf{G}}$ være lineært uafhængige, hvilket er ensbetydende med den lineære transformation i form af matrix multiplikationen med $\overline{\mathbf{G}}$ er injektiv. [Pretzel, 1992]

Idet matricen, som beskriver en lineær transformation, dannes udfra hvordan transformationen påvirker enhedsvektorerne, kan generatormatricen opskrives, som:

$$\begin{aligned} \overline{\mathbf{G}} &= \begin{bmatrix} E(\mathbf{e_1}) & E(\mathbf{e_2}) & \cdots & E(\mathbf{e_k}) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I_1} & \mathbf{I_2} & \cdots & \mathbf{I_k} \\ \mathbf{P_1} & \mathbf{P_2} & \cdots & \mathbf{P_k} \end{bmatrix} \end{aligned} \tag{4.4}$$

Disse paritets søjlevektorer $[\mathbf{P_1} \cdots \mathbf{P_k}]$ bestemmer kodens egenskaber, idet de afgør Hamming vægten af de kodede vektorer. Derved fastlægger paritetssøjlevektorerne kodens d_{min}.

Ud fra en ønsket d_{min}, er det således muligt at bestemme paritets søjlevektorerne igennem iterationer, hvor alle lineært uafhængige vektorer med dimensionen r og Hamming vægt større end $d_{min} - 1$ gennemprøves. Hvis iterationen ikke giver noget resultat, sænkes d_{min} og iterationen foretages igen.

Når disse paritets søjlevektorer er bestemt, er den lineære afbildning fra $\mathbf{B}^k$ over i $\mathbf{B}^n$ fastlagt, idet generator matricen $\overline{\mathbf{G}}$ kendes.

Dekodning. Når en datavektor $\mathbf{R}$ med længden n modtages, skal det undersøges, om den er lovlig, hvilket vil sige om den tilhører værdimængden (range) for den lineære transformation E. Når den lineære transformation er beskrevet ved en matrix multiplikation, vil værdimængden for transformationen være søjlerummet af den pågældende matrix. [Lay, 1996]

Det skal således undersøges, om:

$$\mathbf{R} \in \mathrm{Col}\left(\overline{\mathbf{G}}\right) \tag{4.5}$$

Hvis ligning (4.5) skal være opfyldt, skal ligningssystemet $\overline{\mathbf{G}}\mathbf{x} = \mathbf{R}$ være konsistent. For at undersøge det, er det nødvendigt at udføre række operationer på systemet $[\overline{\mathbf{G}} \quad \mathbf{R}]$.

En lettere måde at undersøge, om ligning (4.5) er opfyldt, er at undersøge om $\mathbf{R}$ står orthogonalt på søjlerummets orthogonale rum, hvilket vil sige:

$$\mathbf{R}^{\perp} \in \mathrm{Col}\left(\overline{\mathbf{G}}\right)^{\perp} \tag{4.6}$$

Rummet, som står orthogonalt på søjlerummet, er L-nulrummet, som er bestemt ud fra:

$$\mathrm{Col}\left(\overline{\mathbf{G}}\right)^{\perp} = \mathrm{Nul}\left(\overline{\mathbf{G}}^T\right) = \{\mathbf{x} : \overline{\mathbf{G}}^T\mathbf{x} = \mathbf{0}\} \tag{4.7}$$

Til beskrivelse af L-nulrummet indføres matricen $\overline{\mathbf{H}}$, hvis søjler udspænder L-nulrummet. Der gælder således, at:

$$\mathrm{Col}\left(\overline{\mathbf{H}}\right) = \mathrm{Nul}\left(\overline{\mathbf{G}}^T\right) = \{\mathbf{x} : \overline{\mathbf{G}}^T\mathbf{x} = \mathbf{0}\} \tag{4.8}$$

Ved kombination af ligning (4.7) og (4.8) fås:

$$\text{Col}\left(\overline{\mathbf{H}}\right) = \text{Col}\left(\overline{\mathbf{G}}\right)^{\perp} \Leftrightarrow \overline{\mathbf{G}} \perp \overline{\mathbf{H}} \Leftrightarrow \overline{\mathbf{H}}^T \overline{\mathbf{G}} = \overline{\mathbf{G}}^T \overline{\mathbf{H}} = \overline{\mathbf{0}} \tag{4.9}$$

Dvs. for at **R** skal være en lovlig kodevektor, skal der gælde:

$$\mathbf{R} \in \text{col}\left(\overline{\mathbf{H}}\right)^{\perp} \Leftrightarrow \overline{\mathbf{H}} \perp \mathbf{R} \Leftrightarrow \overline{\mathbf{H}}^T \mathbf{R} = \mathbf{0} \tag{4.10}$$

Det er således muligt ud fra ligning (4.10) at afgøre, om en modtaget datavektor er gyldig, idet lovlige kodevektorer multipliceret med den transponerede check matrix ($\overline{\mathbf{H}}^T$) altid vil give nulvektoren. Produktet kaldes syndromet for den modtagede datavektor og betegnes **S**

$$\mathbf{S} = \overline{\mathbf{H}}^T \mathbf{R} \tag{4.11}$$

Den modtagede datavektor **R** antages efterfølgende at være summen af en gyldigt datavektor **C** og en fejlvektor **E**. Ved indsættelse af dette i ligning 4.11, samt ved benyttelse af at matrix multiplikation er associativt, fås:

$$\begin{aligned} \mathbf{S} &= \overline{\mathbf{H}}^T \cdot (\mathbf{C} \oplus \mathbf{E}) \\ &= \overline{\mathbf{H}}^T \cdot \mathbf{C} \oplus \overline{\mathbf{H}}^T \cdot \mathbf{E} \\ &= \overline{\mathbf{H}}^T \cdot \overline{\mathbf{G}} \cdot \mathbf{D} \oplus \overline{\mathbf{H}}^T \cdot \mathbf{E} \\ &= \overline{\mathbf{H}}^T \cdot \mathbf{E} \end{aligned} \tag{4.12}$$

Dvs. hvis $\mathbf{E} = \mathbf{0}$ svarende til, at der ikke er sket fejl under transmissionen, vil $\mathbf{S} = \mathbf{0}$. Er syndromet derimod forskellig fra 0, er der sket en eller flere fejl. Hvis der er sket én fejl, svarende til at **E** har Hamming vægten 1, vil syndromet være lig en af søjlerne i $\overline{\mathbf{H}}^T$. Hver søjle svarer således til én diskret fejlposition, hvorved fejlen kan rettes ved en negering.

4.2.4 Bestemmelse af (16,11) kode

Ud fra protokollen er der valgt en samlet datavektor længde på 16-bit (2 byte) med 11 informationsbit og 5 paritetsbit. Den type kode kaldes også en udvidet Hamming kode og har $d_{min} = 4$. Denne kode giver mulighed for at rette én fejl eller detektere to fejl. Richard W. Hamming påviste, at en sådan kode eksisterer, hvorfor det efterfølgende er ønskeligt at få defineret koden i form af generator og check matrix [Pretzel, 1992]. Dette gøres vha. iterationer i MatLab.

Indledningsvis bemærkes, at paritetssøjlevektorerne som beskrevet ved ligning (4.4), alle skal have en Hamming vægt større end 2, for at enhedsvektorerne bliver kodet korrekt (med Hamming vægt 4). Dette skyldes, at søjlerne i generatormatricen er lovlige kodede datavektorer, hvorfor disse nødvendigvis skal have en Hamming vægt større end 4, for at koden har $d_{min} = 4$.

Paritetssøjlevektorerne skal således findes blandt vektorerne, der har længde 5 og vægt større end 2. MatLab løkken beskrevet ved listning 4.1 finder ud fra de 31 forskellige 5-bit vektorer, de 16 vektorer, som har en Hamming vægt større end 2. Ud af disse 16 skal 5 søjlevektorer fravælges for at opnå de korrekte antal.

```
x=5;
for i = 1:(2^x-1);
    w=0;
    P = de2bi(i,x);
    for j = 1:x;
        w=w+P(1,j);
    end;
    if (w>2);
        A=[A;i];
    end;
end;
```

Listing 4.1: Bestemmelse af 5-bit datavektorer med Hamming vægt >2.

Ud af en mængde på 16, er der følgende muligheder for at fravælge 5 elementer, når der ikke tages hensyn til rækkefølgen:

$$\begin{pmatrix} 16 \\ 5 \end{pmatrix} = \frac{16!}{5!(16!-5!)} = 4368 \tag{4.13}$$

Til udvælgelse af disse 4368 forskellige kombinationer genereres en matrix med 4368×16 elementer, hvor hver række repræsenterer en 16-bit datavektorer med Hamming vægt på 11. Dette gøres vha. MatLab listingen 4.2.

```
x=16;
for i = 1:(2^x-1);
    w=0;
    P = de2bi(i,x);
    for j = 1:x;
        w=w+P(1,j);
    end;
    if (w==11);
        B=[B;i];
    end;
end;
```

Listing 4.2: Bestemmelse af 16-bit datavektorer med Hamming vægt 11.

Når matricerne A (indeholdende 5-bit ord med vægt > 2) og B (indeholdende 16-bit ord med vægt = 11) kendes, er det muligt vha. MatLab (listing 4.3) at gennemløbe alle mulige generator matricer, samt for hver at udregne deres d_{min}. Når generator matricen med $d_{min} = 4$ findes, gemmes den, hvorefter paritetscheck matricen H udregnes.

```
for i = 1:length(B);
    S = de2bi(B(i,1),16);   % Udvælg det i. 16-bit/11-vægts ord
    G = A.* transpose(S);   % Gange overflødige elementer ud.
    K = 0;
    for j = 1:16;           % Fjern nul elementer
        if (G(j,1)~=0);
            K = [K;G(j,1)];
        end;
    end;
    K(1,:)=[];              % Paritetsmatrix er bestemt
    G=de2bi(K);
    G = [H G];              % Definer generator udfra paritet og I11x11
    dmin = 100;
    for j = 1:(2^11-1);     % Udregn kodens dmin
        C = de2bi(j,11);    % C er ukodet 11-bit datavektor
        D = mod((C*G),2);   % D er modulo-2 produktet CG
        w = 0;
        for k = 1:16;       % Udregn D's vægt
            w = w + D(1,k);
        end;
        if (w < dmin);      % If løkke finder mindste vægt for G
            dmin = w;
        end;
    end;
    if (dmin == 4);         % Hvis dmin = 4 -> Korrekt generator gemmes
        save generator G;
    end;
end;
G = transpose(G)
H = null(transpose(G),'r'); % Udregn H som nulrummet af G transponeret
HT = transpose(mod(H,2));
```

Listing 4.3: MatLab program til bestemmelse af generator matrix med d_{min}

Efter gennemløbet af MatLab programmet vist ved listning 4.3, er generator og paritets-check matricerne bestemt ved ligning (4.14) og (4.15)

$$\overline{\mathbf{G}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{4.14}$$

$$\overline{\mathbf{H}}^T = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.15}$$

4.2.5 Implementering i PIC

I henhold til beskrivelsen i afsnit 4.2.4 omtales i det følgende, hvordan Hamming kodning og dekodnings rutinerne er implementeret i programstrukturen.

Kodning. Inden data transmitteres, bliver det Hamming kodet i henhold til generatormatricen - ligning (4.14). Udfra generator matricen aflæses følgende ligninger for de forskellige paritetsbit.

$$P_1 = D_1 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6 \oplus D_8 \oplus D_{11} \tag{4.16}$$

$$P_2 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} \tag{4.17}$$

$$P_3 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} \tag{4.18}$$

$$P_4 = D_2 \oplus D_3 \oplus D_4 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \tag{4.19}$$

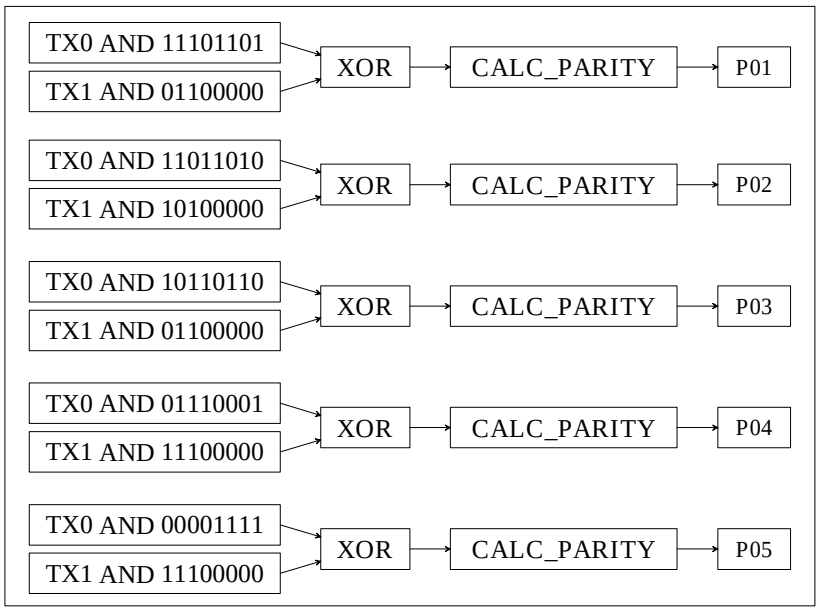
$$P_5 = D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \tag{4.20}$$

De ukodede data ligger i variablerne TX0 og TX1, som vist ved ligning (4.21) og (4.22).

$$\text{TX0} = [D_{01} \quad D_{02} \quad D_{03} \quad D_{04} \quad D_{05} \quad D_{06} \quad D_{07} \quad D_{08}] \tag{4.21}$$

$$\text{TX1} = [D_{09} \quad D_{10} \quad D_{11} \quad xxx \quad xxx \quad xxx \quad xxx \quad xxx] \tag{4.22}$$

Ud fra disse variable (TX0 og TX1), beregnes de 5 paritetsbit i henhold til ligningerne (4.16) til (4.20). Programstrukturen i beregningen af paritetsbittene er vist på figur 4.6.



Figur 4.6: Bestemmelse af paritetsbit.

Efterfølgende, når de 5 paritetsbit er beregnet, sættes disse i variablen TX1. Derved får TX1 strukturen, som vist ved ligning (4.23).

$$\text{TX1} = [D_{09} \quad D_{10} \quad D_{11} \quad P_{01} \quad P_{02} \quad P_{03} \quad P_{04} \quad P_{05}]$$

I programstrukturen for kodningsrutinen benyttes proceduren CALC_PARITY, som beregner pariteten af en byte. Denne procedure er vist ved listing 4.4. Indgangsvariablen Parity_Calc_Data er den byte, for hvilken pariteten ønskes bestemt. Pariteten af byten gemmes i bit 0 i Parity_Calc_Data.

```
CALC_PARITY
    swapf   Parity_Calc_Data, w ; w = D5,D6,D7,D8,D1,D2,D3,D4
    xorwf   Parity_Calc_Data, f ; f = D1+D5,D2+D6,D3+D7,D4+D8,
                                ;     D5+D1,D6+D2,D7+D3,D8+D4
    rrf     Parity_Calc_Data, w ; w = xxxxx,D1+D5,D2+D6,D3+D7,
                                ;   = D4+D8,D5+D1,D6+D2,D7+D3
    xorwf   Parity_Calc_Data, f ; f = D1+D5+xxxx ,D2+D6+D1+D5,
                                ;     D3+D7+D2+D6,D4+D8+D3+D7,
                                ;     D5+D1+D4+D8,D6+D2+D5+D1,
                                ;     D7+D3+D6+D2,D8+D4+D7+D3
    btfsc   Parity_Calc_Data, 2 ; Test pariteten af 2. bit
    incf    Parity_Calc_Data, f ; Toggle bit 0
    return
```

Listing 4.4: Procedure til bestemmelse pariteten for en byte.

Dekodning. Når data modtages, gemmes disse data i variablerne RX0 og RX1, som har strukturen (såfremt der ikke er sket fejl) vist i ligningerne (4.24) og (4.25).

$$\text{RX0} = [D_{01} \quad D_{02} \quad D_{03} \quad D_{04} \quad D_{05} \quad D_{06} \quad D_{07} \quad D_{08}] \tag{4.24}$$

$$\text{RX1} = [D_{09} \quad D_{10} \quad D_{11} \quad P_{01} \quad P_{02} \quad P_{03} \quad P_{04} \quad P_{05}] \tag{4.25}$$

Ud fra disse variable (RX0 og RX1) beregnes syndromet af den modtagne datavektor. Ud fra paritetscheck matricen vist i ligning 4.15, aflæses følgende syndrombit:

$$S_1 = D_1 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6 \oplus D_8 \oplus D_{11} \oplus P_1 \tag{4.26}$$

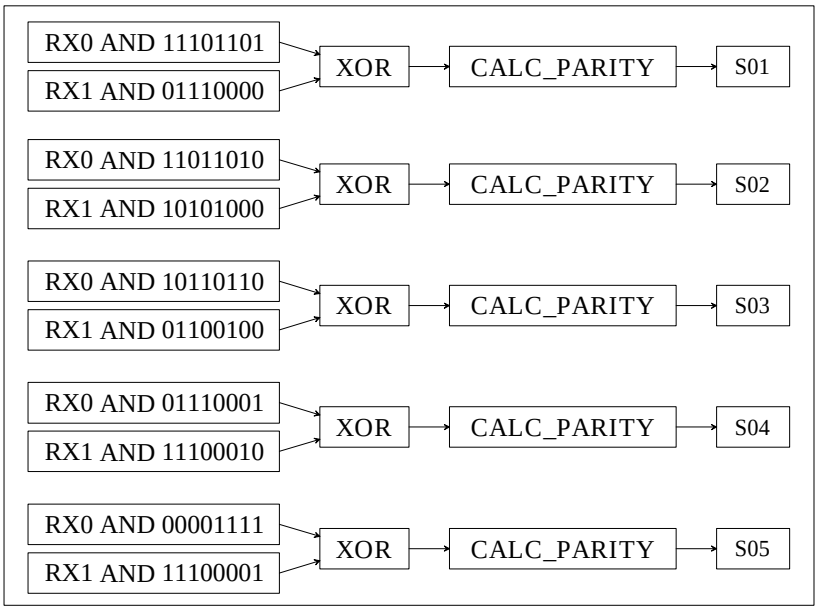
$$S_2 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} \oplus P_2 \tag{4.27}$$

$$S_3 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} \oplus P_3 \tag{4.28}$$

$$S_4 = D_2 \oplus D_3 \oplus D_4 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus P_4 \tag{4.29}$$

$$S_5 = D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus P_5 \tag{4.30}$$

Programflowet i beregningen af syndromvektoreren er vist på figur 4.7.



Figur 4.7: Bestemmelse af syndrombit

Fejlkorrektion. Når syndromet ud fra ligningerne (4.26) til (4.30) beregnes til en værdi forskellig fra nul, betyder det, at der er sket én eller flere fejl. Som nævnt tidligere vil syndromets værdi i forbindelse med én fejl, være lig en af søjlerne i $\overline{\mathbf{H}}^T$, hvorfor det således er ønskeligt at sammenligne den beregnede syndromvektor med paritetscheckmatricen.

For at nedbringe kompleksiteten af denne sammenligningsproces, gøres følgende observation, nåt det bemærkes, at den 1. paritetsbit indeholder pariteten af hele den modtagede datavektor:

$$\begin{aligned} P &= D_1 \oplus \cdots \oplus D_{11} \oplus P_1 \oplus \cdots \oplus P_5 && (4.31) \\ &= D_1 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6 \oplus D_8 \oplus D_{11} \\ &= P_1 && (4.32) \end{aligned}$$

At ligning (4.31) er lig (4.32), ses ved indsættelse af ligningerne (4.16)-(4.20) i ligning (4.31), med efterfølgende reduktionen i henhold til regnereglerne for modulo-2 aritmetik.

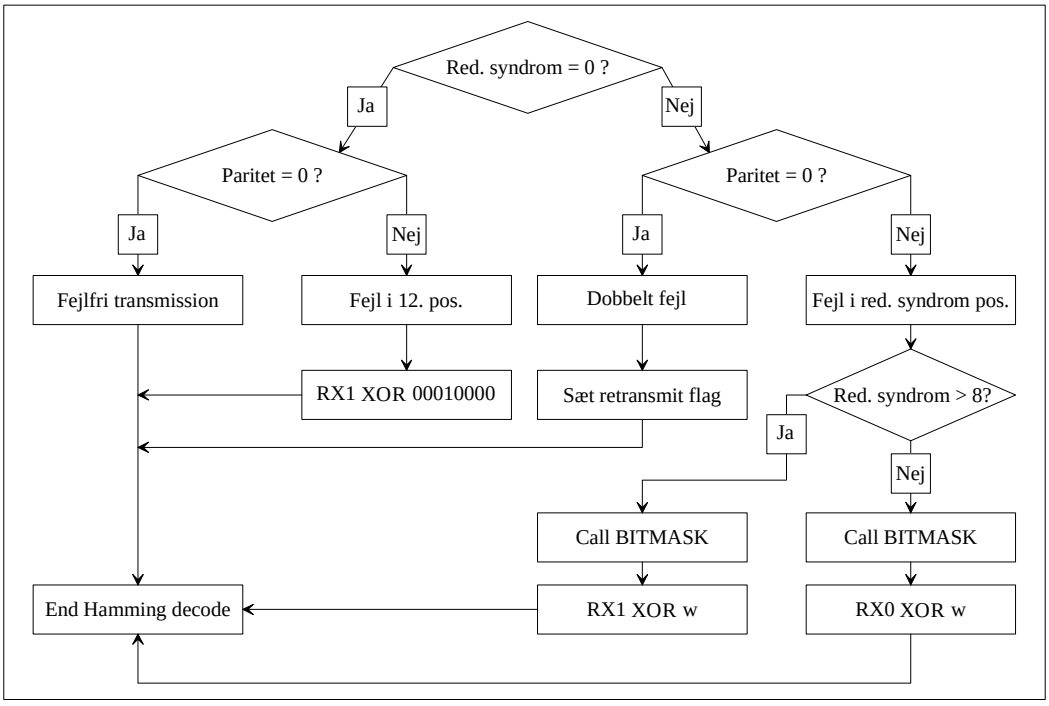
Der indføres dernæst en reduceret syndromvektor, givet ved S_2 til S_5. Ud fra den reducerede syndromvektor, samt pariteten for den modtagede datavektor, er det således muligt at afgøre, om der er sket én eller to fejl. Sker der et ulige antal fejl, vil den modtagede datavektors paritet være ulige. Dvs., sker der én fejl, vil pariteten være ulige, og fejlpositionen

	Syndromet $= 0$	**Syndromet $\neq 0$**
Lige paritet	Fejlfri	Dobbelt fejl
Ulige paritet	Fejl i pos. 12	Fejl i syndrom pos.

Tabel 4.1: Mulige dekodnings udfald

vil entydigt være givet ud fra de resterende syndrom bit (S_2 til S_5). Sker der 2 fejl, vil det reducerede syndrom stadig være forskellig fra nul, mens pariteten vil være lige - derved kan de 2 fejl detekteres. Disse forskellige dekodnings muligheder er opsummert i tabel 4.1.

Når syndromet og pariteten af den modtagne datavektor er bestemt, foretages en handling i henhold til tabel 4.1. Den mere præcise struktur for dekodningsprogrammet er vist på figur 4.8.



Figur 4.8: Programflow for korrektionsprocedure til Hamming dekodning.

Hvis der er detekteret én fejl, benyttes proceduren BitMask til at bestemme en toggle bit. Dvs., hvis syndromet er 4, returnes konstanten 00010000, som derefter kan benyttes til at toggle den forkerte bit i den modtagne datavektor vha. en XOR operation.

4.3 Transmit

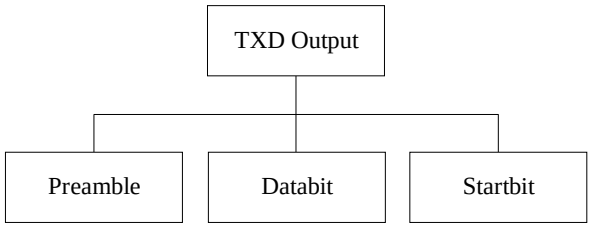
Transmit softwaremodulet er i stand til at sende data mellem delsystemerne. Denne funktion kan udføres ved et kald af den tilhørende rutine RDOSend bestående af flere subrutiner. Ved kald af RDOSend tildeles lokale variable startværdier og et timerbaseret interrupt initialiseres. Timeren indstilles, så den valgte bithastighed opnås.

Som indgangsvariable har funktionen to databytes benævnt TX0 og TX1, der udgør datapakken. TX0 og TX1 er tildelt hukommelsesplads i begge MCU.

Protokollen specificerer, at modulet forud for hver transmission skal sende en preamble på >94 [ms] efterfulgt af to høje startbits. Endvidere manchesterkodes datapakken under afsendelse.

4.3.1 Programstruktur

Transmit softwaremodulet bygges op omkring en subrutine, der kan udlæse data på radiomodulets TXD-ben. Omkring denne subrutine opbygges andre subrutiner. Programstrukturen fremgår af figur 4.9:



Figur 4.9: Programstruktur for transmitprocedure.

TXD Output. Denne subrutine kan ud over at udskrive på TXD benet skifte tilstanden på benet ud fra benets foregående værdi. TXD Output benyttes til at sende preamblens bitmønster og manchesterkode samt sende datapakken. Kildekoden for modulet ses i listing 4.5:

```
TestTXD:        btfss   PORTB,RDOTXD    ;Test TXD
                goto    TXDSet          ;Skift tilstand på TXD (0 til 1)
                goto    TXDClr          ;Skift tilstand på TXD (1 til 0)

TXDSet:         bcf     INTCON,T0IF     ;Clear TMR0 flag
                bsf     PORTB,RDOTXD    ;Sæt TXD høj
                goto    ENDINT

TXDClr:         bcf     INTCON,T0IF     ;Clear TMR0 flag
                bcf     PORTB,RDOTXD    ;Sæt TXD lav
                goto    ENDINT
```

Listing 4.5: Kildekode for TXD Output.

Preamble. Varigheden T af preamblen vælges indledningsvist til at være ca. 110 [ms]. Antallet af bit N, der sendes i dette tidsrum T, er givet ved:

$$N = B \cdot T = 9600[s^{-1}] \cdot 0.110[s] = 1.06 \cdot 10^3 \tag{4.33}$$

hvor B er bithastigheden, der i dette tilfælde er den samme som baudhastigheden valgt til 9600 [baud]. For at sende 1060 bit indføres en variabel preampc med startværdi N, der tæller ned, for hver bit der sendes. Når værdien af variablen er nul, er preamblen sendt.

Da hukommelsescellerne i de valgte MCU er en byte lang, benyttes to hukommelsesceller preampc1 og preampc2. Preampc2 nedtælles med 1, for hver gang preampc1 nedtælles med 256. Startværdien for preampc er da:

$$preampc = preampc1 \cdot preampc2 \tag{4.34}$$

idet startværdien for preampc1 er 255+1. Implementeret i programkoden svarer denne værdi til en tom fil. Der kan af formel 4.34 bestemmes en værdi for preampc2:

$$preampc2 = \frac{preampc}{preampc1} \tag{4.35}$$

$$= \frac{1060}{256} \approx 4 \tag{4.36}$$

Preamblens varighed er med denne værdi:

$$T = \frac{N}{B} = \frac{preampc1 \cdot preampc2}{B} = \frac{256 \cdot 4}{9600[baud]} = 107[ms] \tag{4.37}$$

hvilket er tilstrækkeligt.

I listing 4.6 ses, hvorledes dette er implementeret i programkoden:

```
Preamp:     decfsz  preampc1,f          ;Tæller preampc1 ned
            goto    TestTXD
            decfsz  preampc2,f          ;For hver nedtælling af preampc1
                                        ;med 256 tælles preampc2 ned med 1
            goto    TestTXD
            bsf     TX_RDC, TX_PREAMBLE ;Sætter kontrolbit
                                        ;til interruptkontrol
            goto    TestTXD
```

Listing 4.6: Kildekode for preamble.

Det bemærkes, at denne subrutine benytter TestTXD fra TXDOutput subrutinen til at skifte tilstand på TXD benet.

Startbit. Den sidste bit, der sendes i preamblen, er logisk lav. Subrutinen StartBit sætter TXD benet logisk højt, svarende til tiden det tager at sende to bits. Dette styres af variablen startbitc, der tæller 1 ned hver gang, der sendes et startbit.

Databit. TXDEncode subrutinen manchesterkoder og udskriver de enkelte bits i datapakken på TXD benet. En variabel sørger for at udskrive et manchesterbit ved hvert andet gennemløb ved kald af TestTXD i TXDOutput, der skifter tilstand på TXD benet:

```
TXDEncode:  decfsz  markspace,f     ;Tæller markspace ned
            goto    TXFileTest      ;Hver 2. gang sendes en bit i TX(x)
            bsf     markspace,1     ;Hver 2. gang sættes markspace til 2
            goto    TestTXD         ;Hver 2. gang sendes et mark/space bit
```

Listing 4.7: Kildekode til styring af manchesterkodning.

Til at udskrive datapakkens to bytes benyttes princippet i inddirekte adressering med en pointer RDO_RXPTR, der peger på den fil, der skal udlæses. Til at kontrollere om den første byte er udlæst, nedtælles en variabel txdatac med startværdi 9. Ved det niende gennemløb øges pointerens værdi med 1 og peger på den sidste byte, der skal udlæses:

```
TXFileTest: decfsz  txdatac,f       ;Tester om TX(x) er udlæst
            goto    TXFileOut       ;- hvis ikke udlæses næste bit
            incf    RDO_RXPTR,f     ;Læg 1 til RDO pointeren
```

Listing 4.8: Kontrol af pointer.

4.3.2 Programflow

Rækkefølgen for afvikling af de enkelte subrutiner i modulet styres af en interruptkontrol subrutine. Subrutinen gemmer endvidere arbejdsregister og statusregister i begyndelsen af

hvert interrupt, og gendanner disse under afslutning af samme interrupt. Efter transmission kaldes en reset subrutine, der tildeler de brugte variable startværdier, og deaktiverer det timerbaserede interrupt.

Til interruptkontrol subrutinen knytter sig et kontrolregister TX_RDC. Dette ses på figur 4.3.2:

Register	7.bit	6.bit	5.bit	4.bit	3.bit	2.bit	1.bit	0.bit
TX_RDC			TX_REQ		TX_OK	TX_START	TX_PREAMBLE	TX_GO

Tabel 4.2: Bits indeholdt i TX_RDC.

Interruptkontrollen kan ved bittest afgøre, hvilken subrutine der skal kaldes. Betydning af de enkelte bits i kontrolregistret beskrives herunder:

- TX_GO: Sættes efter initialisering af RDOSend. Ved næste interrupt påbegyndes transmission af preamble. Benyttes af hovedprogram til at teste for igangværende transmission.
- TX_PREAMBLE: Sættes efter preamble er sendt. Ved næste interrupt påbegyndes transmission af startbit.
- TX_START: Sættes efter startbits er sendt. Ved næste interrupt påbegyndes transmission af datapakke.
- TX_OK: Sættes efter datapakke er sendt. Er benyttet under programudviklingen.
- TX_REQ: Sættes af andre softwaremoduler, der ønsker at sende en datapakke. Kontrol af TX_REQ og kald af RDOSend styres af hovedprogrammet.

4.4 Receive

Princippet i modtagelsen af radiodata, minder meget om princippet fra afsnit 4.3 om datatransmission. De overordnede punkter ridses op her:

1. Opstart af RDO.
2. Undersøg om der findes bærebølge på den benyttede frekvens (433.920 [MHz]).
3. Synkronisér og find startbit, hvis det er muligt.
4. Modtag og Manchesterdekod data.
5. Signalér at data er klar, og om de er modtaget uden fejl.

Opstart af RDO. Opstarten sker som følge af et procedurekald fra main-løkken (tidligere beskrevet i kapitel 3.3). Main-løkken starter kun radiomodtagelsen, hvis modulet ikke i forvejen er ved at sende, eller der endnu ligger ubehandlede data klar fra en tidligere datamodtagelse. Dette kan i PIC kode se ud som i listing 4.9:

```
PROCEDURE START

btfsc  RX_RDC, RX_CD       ;Returnér hvis RDO i forvejen modtager
return

btfsc  TX_RDC, TX_GO       ;Returnér hvis RDO i forvejen sender
return

btfsc  RX_RDC, RX_OK6BYTE  ;Returnér hvis tidligere data findes
return

OPSTART af RDO
```

Listing 4.9: Betinget opstart af radiomodtagelse.

Kontrolregistrene, som anvendes i forbindelse med radiokommunikationen (RX_RDC og TX_RDC), består af følgende elementer:

Register	7.bit	6.bit	5.bit	4.bit	3.bit	2.bit	1.bit	0.bit
RX_RDC	RX_ERRMAN	RX_ERRTMOUT	RX_ERRCD	RX_OK6BYTE		RX_START	RX_SYNC	RX_CD
TX_RDC			TX_REQ		TX_OK	TX_START	TX_PREAMBLE	TX_GO

Tabel 4.3: Bits indeholdt i RX_RDC og TX_RDC.

Registret TX_RDC er defineret i forbindelse med afsnit 4.3 men benyttes også under modtagelse, idet det fremgår af registret om radiomodulet er ved at sende. Betydningen af de forskellige flag i RX_RDC kan kort sammenfattes som:

- RX_ERRMAN: Manchester dekodningsfejl
- RX_ERRTMOUT: Radio timeout
- RX_ERRCD: Bærebølgen gik tabt før sidste bit var modtaget
- RX_OK6BYTE: Alle data er modtaget
- RX_START: Startbitmøsnter genkendt
- RX_SYNC: Modtager synkroniseret med afsender
- RX_CD: Bærebølge eksisterer

Overholdes startbetingelserne, hvad registrene angår, vækkes radiomodulet fra standby. Det er herefter, i følge databladet for radiomodulet [Radiometrix, 2001], nødvendigt at vente ca. 2 [ms] på at CD (Carrier Detect) bliver stabil. I programmet til MCU er der indbygget en sikkerhedsmargin, og delayet er således specificeret til at vare 5 [ms].

Er $\overline{CD}$ benet på radiomodulet højt, betyder det, at der ikke kan opfanges en bærebølge, og radiomodulet slukkes. Er benet derimod lavt, sættes RX_CD flaget i kontrolregistret, og et timerbaseret interrupt startes.

Synkronisering og startbit. For at kunne synkronisere modtageren med afsenderen, er det valgt at sample data fra radiomodulet med tre gange oversampling. Dette gøres ved at sætte tiden mellem hvert modtagende interrupt til en trediedel af tiden, som senderen benytter mellem sine interrupts, hvilket ved 9600 [baud] vil sige:

$$\frac{1}{3} \cdot T_{TX} = T_{RX} \Rightarrow \frac{1}{3} \cdot \frac{1}{9600[baud]} = \frac{1}{28800[baud]} \approx 34.7[\mu s] \tag{4.38}$$

I (4.4) er T_{RX} og T_{TX} henholdsvis periodetiden mellem modtagende og afsendende interrupts.
Når timerinterruptet opstår, roteres den tilstand, der måtte være på radioens databen (RXD) ind i et register fra højre. Dette register sammenholdes med en, for processoren kendt synkroniseringsnøgle, og hvis registret matcher nøglen, sættes RX_SYNC, som angiver, at modtageren er synkroniseret.
Synkroniseringsnøglen kan opskrives ud fra eksempel 4.4.

Afsendt byte	**Modtaget byte**	**Nøgle**
01010101	00011100 01110001 11000111	1 1100011
10101010	11100011 10001110 00111000	0 0011100

Tabel 4.4: Eksempel på synkroniseringsnøgle ved tre gange oversampling.

Årsagen til, den sidste bit ikke er med i nøglen, er, at synkroniseringen på denne måde opnås midt i databitten i stedet for på kanten af bitten. Processorens samplehastighed sættes efter synkronisering til 9690 [baud] for at aflaste processoren, så den kan udføre flere instruktioner mellem hvert interrupt. Antallet af instruktioner mellem hvert interrupt før og efter synkronisering kan, når én instruktion tager 200[ns], beregnes som:

$$\text{INT}\left(\frac{1}{29070[baud] \cdot 200[ns]}\right) = 172 \tag{4.39}$$

$$\text{INT}\left(\frac{1}{9690[baud] \cdot 200[ns]}\right) = 516 \tag{4.40}$$

Af disse instruktioner afvikles en del i selve interruptet, og resten kan således bruges på at opdatere display, undersøge knaptilstande, etc.

Programmet i microprocessoren indeholder ligeledes en procedure til at finde startbittene. Den fungerer, som synkroniseringsalgoritmen, ved hjælp af en nøgle og efter samme princip. Når startbitten er fundet, sættes RX_START høj, hvilket medfører, at processoren starter den egentlige dataindlæsning.

Modtag og Manchesterdekod data. Da alle databits er Manchesterkodet, roteres to databit ind i et midlertidigt register, hvorefter der ses på den mest betydende bit (MSb). Er

MSb høj, må LSb skulle være lav, såfremt synkroniseringen er korrekt. Er begge bits ens, er der sket en Manchesterdekodningsfejl og flaget RX_ERRMAN sættes. Dette betyder i praksis, at LSb tilrettes, således den er modsat af MSb. Denne proces fortsætter indtil alle bits er indskiftet, eller der opstår en uoprettelig fejl. En sådan fejl optræder f.eks., hvis CD mistes, før alle databits er modtaget, og medfører at fejlflaget RX_ERRCD sættes, samt at radiokommunikationen termineres.
Er derimod alle databits indskiftet, sættes flaget RX_OK6BYTE, og radiomodulet overgår til standby mode. Kontrolregistrets tre laveste bits (RDO relateret) nulstilles, mens de øvrige (datarelateret) beholdes til videre behandling.

Næste gang main-løkken gennemløbes, vil RX_OK6BYTE flaget være sat, og de nødvendige procedurer til Hammingdekodning samt videre behandling udføres.

4.5 Opkode

Kommunikationen mellem de to microprocessorsystemer, fungerer ved hjælp af et system af opkoder (operations koder). Længden af opkoderne er bestemt af protokollen. Den valgte Hammingkode anvender som nævnt fem bits til kontrol, hvilket betyder, at de resterende tre bits (7., 6. og 5. bit) med fordel kan anvendes til opkoder. Den sidste databyte kan så anvendes til at overføre egentlige data, som f.eks. en temperatur.

En generel måde at nedbryde opkoder på, så processoren kan forstå, hvad der skal udføres fremgår af tabel 4.5:

Er 7.bit høj	Er 6.bit høj	Er 5. bit høj	$\rightarrow$	Opkode = 111
		Er 5. bit lav	$\rightarrow$	Opkode = 110
	Er 6.bit lav	Er 5. bit høj	$\rightarrow$	Opkode = 101
		Er 5. bit lav	$\rightarrow$	Opkode = 100
Er 7.bit lav	Er 6.bit høj	Er 5. bit høj	$\rightarrow$	Opkode = 011
		Er 5. bit lav	$\rightarrow$	Opkode = 010
	Er 6.bit lav	Er 5. bit høj	$\rightarrow$	Opkode = 001
		Er 5. bit lav	$\rightarrow$	Opkode = 000

Problemet med metoden er, som det ses, at der sker en forgrening af rutiner, som skal teste de involverede bits. Hvis der f.eks. var brugt otte bits i opkoden, ville det kræve ialt 2^8 (256) forskellige bittest rutiner at afkode betydningen af en instruktion. Det er besværligt at kode, og endnu mere besværligt at overskue, hvorfor der er anvendt en bedre metode.
Afkodningen sker istedet ved, at de tre bits (7., 6. og 5. bit) flyttes ned som de tre nederste bits i en byte (2., 1. og 0.bit), hvorefter de adderes til en basisadresse i PICens ROM. Resultatet gemmes i PC (Program Counter) og dette får processoren til at hoppe til den pågældende adresse. Assembleren underrettes, ved hjælp af ORG kommandoen, om at 2^3 (8) konsekutive adresser i forlængelse af basisadressen er reserveret til opkoder. Hver opkode

kan nu fylde én program linie, i en specifikt angiven adressecelle. Cellerne indeholder hver en goto kommando, som får processoren til at udføre den kommando, som opkoden skal afstedkomme.
Dekodningsproceduren er meget afhængig af, at der ikke er andre procedurer, som lægger beslag på samme ROM-område, men fungerer hurtigt og er nem at overskue. Listing 4.10 viser programkoden til omtalte procedure.

```
clrf   PCLATH      ;Slet øverste bits i PC

movf   RX1, W      ;Kopier modtagen opkode ind i W
andlw  b'11100000' ;Hvis der er fejl, behold kun 3 mest betydende bits
movwf  RX1         ;Læg det nye resultat af opkoden tilbage i filen

bcf    STATUS, C   ;Sæt carry til nul (vigtig ved højreskift)
swapf  RX1, F      ;Byt nybbles
rrf    RX1, F      ;Sidste højreskift, så bits 7 til 5 ligger i 2 til 0
movf   RX1, W      ;Kopier resultatet ind i W
clrf   RX1         ;Slet den oprindelige fil

addlw  0Ah         ;Læg basisadressen 0Ah til opkoden
movwf  PCL         ;Gå til pågældende ROM-adresse (0Ah til 11h)
```

Listing 4.10: Fortolkning af opkoder i assemblerkode.

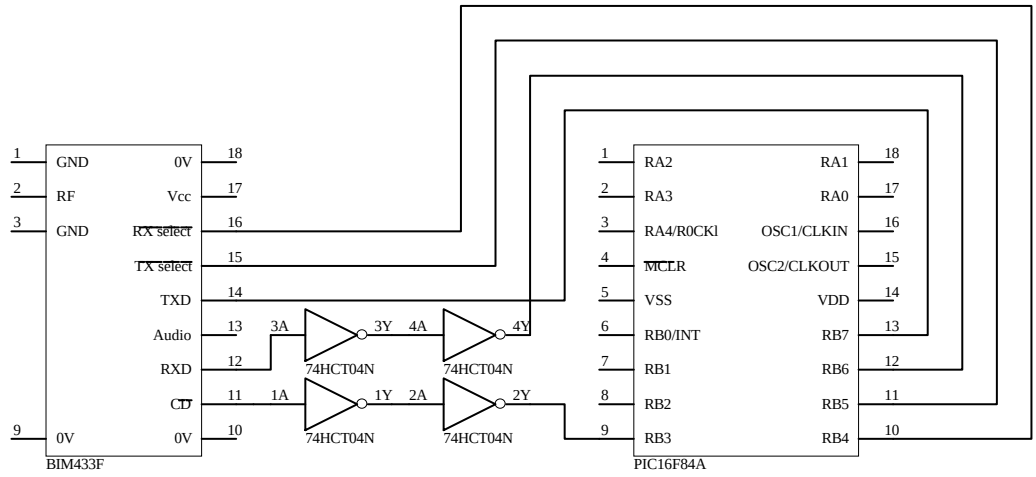
Den egentlige kommunikation mellem de to microprocessorsystemer fungerer overordnet således:

Mastersystem	**Slavesystem**
Opkode vælges og kodes.	
Opkode afsendes.	
	Opkode dekodes.
	Handling jf. opkode udføres (f.eks. temp. måling).
	Opkode vælges og kodes sammen med evt. data.
	Data og opkode afsendes.
Opkode dekodes.	
Handling udføres (f.eks. udskrift på display).	

Under alle normale operationer, er fjernbetjeningen (PIC16F84A) master, og bliver kun slave, hvis bilen (PIC16F877) detekterer en alarmkondition.

4.6 Hardware grænseflade mellem radiolink og MCU

Dette afsnit beskriver hardwaren, der ligger til grund for kommunikationen mellem radiolink og MCU. På figur 4.10 ses sammenkoblingen af de to delsystemer, som her er isoleret fra resten af systemet.



Figur 4.10: Grænseflade mellem P16F84A og BIM-433-F.

For både PIC16F84A og PIC16F877 er radiomodulet koblet til RB3 til RB7 på PORTB, der har TTL interface. RB3 og RB6 er sat til at modtage input fra CD og RXD benene på BIM-433-F. CD benet har en udgangsimpedans på 50 [kΩ] og er kun beregnet til CMOS interface. RXD benet har en udgangsimpedans på 10 [kΩ], hvorfor databladet for BIM-433-F anbefaler et CMOS interface på dette ben.

Ydermere er de interne pull up modstande på 10 [kΩ] aktiveret på input benene på PORTB. Spændingsdelingen mellem pull up modstandene og udgangsmodstandene på CD og RXD benene betyder da, at hverken CD eller RXD benene med sikkerhed kan trække RB3 og RB6 logisk lav.

CMOS interface på CD og RXD benene opnås ved at indsætte en TTL kompatibel CMOS hexinverter 74HCT04N fra Phillips. Det fremgår af figur 4.10, at spændingen på de to ben inverteres to gange, hvormed den ønskede impedanstilpasning og indgangsspænding på RB3 og RB6 opnås. Hexinverteren drives af forsyningsspændingen. Ved en forsyningsspænding på $V_{CC} = 4.5[V]$ og en temperatur på $T_{amb} = 25[°C]$ specificerer databladet et propagation delay på $t_{PHL} = t_{PLH} \leq 19[ns]$ og en transition time på $t_{THL} = t_{TLH} \leq 15[ns]$. Dette har ikke nogen indflydelse på synkroniseringen af de to radiomoduler og på systemets funktionalitet iøvrigt.

4.7 Delkonklusion

Ud fra den opstillede protokol er der således udviklet softwaremoduler der gør radiomodulerne i stand til at sende og modtage data med den ønskede hastighed. Endvidere er der udviklet kodnings og dekodnings softwaremoduler der kan rette én fejl eller detektere to fejl under transmission.

Disse moduler udgør, sammen med opkode-systemet, en tilfredsstillende kommunikationsstandard til overførsel af data som omtalt i kravsspecifikationen.

Brugerinterface 5

Formålet med dette kapitel er at beskrive den grænseflade, som eksisterer imellem brugeren og master systemet. Dette indebærer en beskrivelse af LCD displayet, som benyttes til udlæsning af data, samt en forklaring af hvordan knaplogikken og piezo transducer er implementeret i mastersystemet.

5.1 Display

Til projektet anvendes et dot matrix karakter display til udlæsning af data til brugeren, omkring temperatur, benzin etc. Dette skal ske på forståelig og logisk måde. Der henvises til appendiks A.1 for uddybet gennemgang af den teknisk implementering samt gennemgang af de vigtigste rutiner, der driver LCD displayet.

I dette afsnit gennemgås, hvorledes rutinerne er opbygget samt hvordan de kan sammensættes til opbygning af menuer og udlæsning af data. Det bliver oplyst hvilke hukommelsesceller de eksterne procedurer benytter, samt hvilke af LCD modulets rutiner der skal kaldes hvornår.

5.1.1 Rutinernes opbygning

LCD modulet kan ses som en helhed, der ved et enkelt kald kan udskrive en skærmmenu. Modulet kan også opdeles i elementer, der hver især kan kaldes, hvis der findes et behov for udlæsning af enkelte kommandoer, display karakter eller et bitmønster.

De enkelte procedurer kan kaldes alt efter, hvad der ønskes udskrevet på displayet og forudsætter ikke, at der er yderligere procedurer, der er kaldt inden.

5.1.2 Input og output

Ved alle kald af lcd modulets rutiner, udlæses outputtet på displayet, medmindre kaldet er beregnet for kommandoer. Alle tegn og kommandoer skal, inden de aktiveres, lægges ind i hukommelsescellen lcd_dataud, hvorefter der kaldes en rutine, der behandler hukommelsescellens data til den specifikke funktion. Ved udskrivning af menuer kaldes menuens navn f.eks. lcd_menu1. Ønskes der udskrevet en hel tekststreng, angives tekststrengens start og

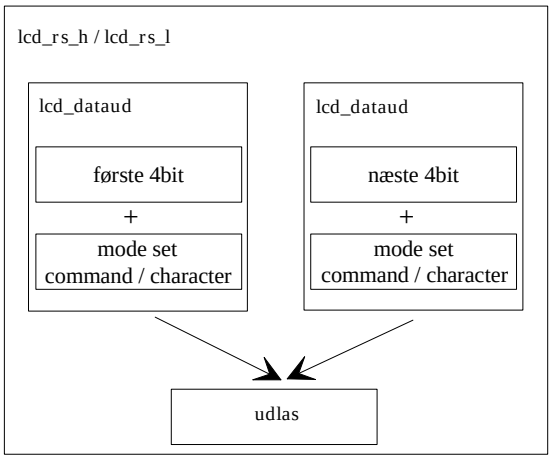
slut position til hukommelsescellerne lcd_tab_beg og lcd_tab_end. Derefter afsluttes med kaldet af lcd_udskriv, der sørger for en gennemgang og tilrettelæggelse af alle tegn der skal udlæses, med henblik på kald af de rette procedurer i den rigtige rækkefølge.

5.1.3 Delelementer i LCD modulet

For at forstå hvorledes alle displayets rutiner arbejder sammen, er det nødvendigt at se på, hvilke funktioner rutinerne hver især udfører, hvorfor en gennemgang af disse foretages:

Udlas. Kald af rutinen udlas sørger for, at uanset hvad der ligger i hukommelsescellen lcd_dataud, sendes dette serielt til displayet, gennem et skifteregister. Det er derfor vigtigt, at de rigtige data ligger klar inden udlas kaldes.

Lcd_rs_h og lcd_rs_l. Hukommelsescellen lcd_dataud er en 8 bit celle. Der benyttes kun 4 bit overførsel til displayet, hvorfor lcd_dataud splittes op og sendes i to portioner. Rutinerne lcd_rs_h og lcd_rs_l sørger for opsplitningen, samt tilføjer tilstanden for RS til displayets controller. Denne tilstand afgør om der udskrives tegn eller kommandoer. Figur 5.1 viser opbygningen af rutinerne.

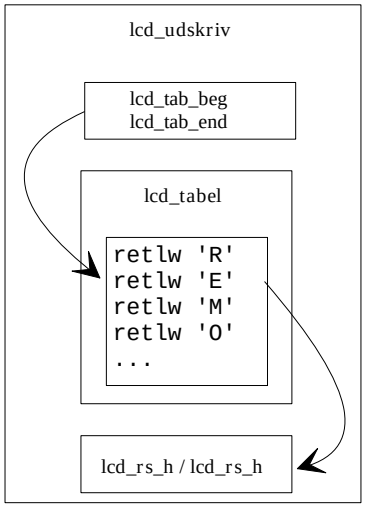


Figur 5.1: lcd_dataud splittes op herefter fortages to udlæsningskald

Figur 5.1 viser, at rutinerne selv sørger for at kalde udlas, når de rette data ligger klar. Hvert kald af lcd_rs_h eller lcd_rs_l udskriver kun et tegn eller en kommando ad gangen.

Lcd_udskriv. Til udskrivning af flere efterfølgende kommandoer eller tegn, vælges der at udskrive fra en tabel, grundet bedre udnyttelse af PICens programkode plads. Rutinen lcd_udskriv kalder tabellen lcd_tabel. Denne henter dataværdien gemt på tabellens position,

samt kalder de respektive underrutiner. Udskrivningen fortsætter, til sidst valgte dataværdi er udlæst.



Figur 5.2: Flowchart af tabel udskrivnings rutinen.

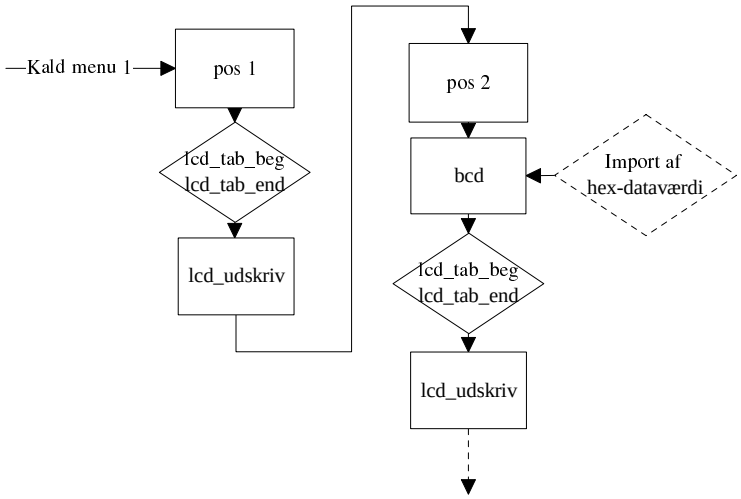
Figur 5.2 viser, hvorledes lcd_udskriv kalder underrutiner. Lcd_tab_beg og lcd_tab_end er variable, der afgør hvor mange tegn, der skal udskrives.

PosX. For at placere displayets cursor på et givet sted, benyttes pos rutinerne. Disse er fast indstillet til at flytte cursoren til forudbestemte steder på displayet.

Bcd. Til udlæsning af decimaltal på displayet er en konvertering fra ASCII til bcd nødvendig. Dette gøres med bcd rutinen, som tager udgangspunkt i, at ASCII værdien er gemt i lcd_dataud inden bcd rutinen gennemløbes. Rutinen konverterer først tallet, hvorefter decimaltallene udskrives, med højeste position først. Det største tal, der kan udskrives, er 255 eftersom lcd_dataud er en 8 bit celle.

5.1.4 Menu opsætning

Ved kald af en menu udskrives tegn og tal valgte steder på displayet. En typisk opsætning af en menu kan foregå efter følgende fremgangsmåde:



Figur 5.3: Typisk opsætning af en menu.

På figur 5.3 ses hvilke rutiner, der skal kaldes ved opsætning af en menu. Ved kald af en menu, f.eks. når én af fjernbetjeningens knapper aktiveres, sørger menurutinen for kald af alle underlæggende rutiner, der skal bruges til udskrivning af den aktuelle menu. Figur 5.3 viser, der skal hentes data udfra, hvis bcd rutinen anvendes. Menuen kan gøres større, hvis der skal udskrives et mere avanceret skærmbillede.

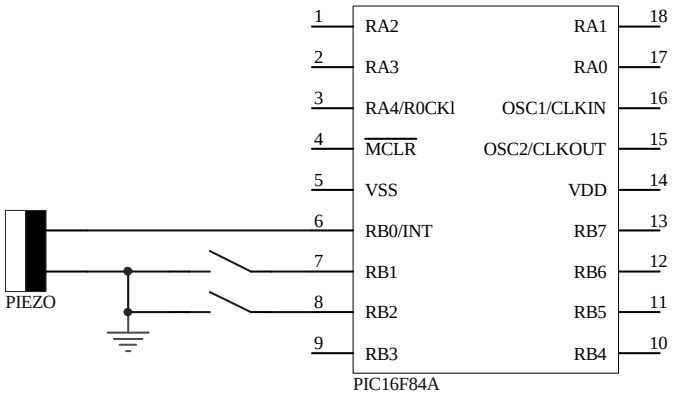
5.2 Knaplogik og piezo transducer

Til interaktion mellem bruger og mastersystemet er implementeret to knapper, hvis funktion er henholdsvis op- og aflåsning af slavesystemet, samt aflæsning af status for temperatur og benzinstand. Knapperne er implementeret i form af trykkontakter, som vist ved figur 5.4. Når kontakterne ikke er sluttet, sørger de interne pull up modstand i PICen for, at RB1 og RB2 ikke svæver. I hvert gennemløb af Main-løkken polles for input på knapperne. Dette er vist ved listing 5.1. Input fra knapperne ignoreres i to tilfælde. Dette sker, enten hvis PICen har afsendt en pakke uden at have modtaget svar, eller hvis der afventer data til afsendelse.

```
KNAP1_POLL:     btfsc   PORTB, KNAP1    ;Hvis KNAP1 er lav,
                                        ;udføres handling
                return

                btfsc   TX_RDC, TX_TIMER;Skip hvis modtagelse
                                        ;mangler
                return

                btfsc   TX_RDC, TX_REQ  ;Ignorér, indtil
                                        ;foregående data er sendt
                return
```

Listing 5.1: Poll rutine til knap 1

Derudover er monteret en piezo transducer til at signalere om eksempelvis alarm i slave-systemet. Transduceren benyttes også til at meddele, om slave systemet er uden for rækkevidde. Den elektriske opkobling er vist ved figur 5.4



Figur 5.4: Elektrisk diagram for opkobling af knapper og piezo transducer

5.3 Delkonklusion

I dette kapitel er det blevet beskrevet, hvordan grænsefladen imellem brugeren og mastersystemet er implementeret. Dette er dels sket igennem en forklaring af LCD modulet, hvor det blevet klargjort, hvad dette modul består af, samt hvorledes delelementerne kan sammensættes til opbygning af menuer. Brugen af menuer, lavet som enkeltstående rutiner, letter overskueligheden af hovedprogrammet betragteligt. Således er det kun nødvendigt at kalde én rutine, for udskrivning af et helt skærmbillede. Eksempelvis er initialisering af displayet opbygget ligesom en menu, blot er det kommandoer og ikke tegn, der udskrives til displayet.

Derudover er det gennemgået, hvordan knaplogikken er opbygget, samt hvordan den er implementeret i form af knapper til op- og aflåsning og aflæsning af status for temperatur og benzin. Endelig er piezo transducerens implementering beskrevet.

Dataopsamling

6

Dette afsnit indeholder implementeringen af de senorer og aktuatorer, der er tilkoblet PIC16F877.

6.1 Benzinmåler

Benzinmåleren bliver emuleret ved et potentiometer. Ved regulering af potentiometeret vil spændingen til A/D converteren ændre sig og derved simulere en benzintanks indholdsniveau. I PIC16F877 er der implementeret en 10 bit A/D converter, der ved hjælp af en kapacitorstige bruger successiv approksimation til at finde frem til den påførte analoge spænding. Ønskes der en større gennemgang af A/D converterens struktur mht. funktionalitet, design og opsætning, henvises der til Appendiks B.

6.1.1 Konfiguration af A/D converteren

Ved initialisering af A/D converteren skal følgende ting overvejes:

- **Porte:** Hvor mange analoge og digitale porte skal bruges.
- **Referencespænding (V_{REF}):** Skal V_{REF} være V_{DD}, eller en specifik valgt spænding på en input port.
- **Valg A/D clock (T_{AD}):** Hvilken periodetid skal T_{AD} have.
- **Interrupt:** Skal A/D konverteringen interrupt styres.

Når disse punkter er overvejet, kan initialiseringen af A/D converteren programmeres.

```
        bsf     STATUS,RP0      ;Reg.Bank1
        movlw   00FFH           ;b'11111111'
        movwf   PORTA           ;PORTA input

        movlw   B'00001110'     ;Resultat venstrejusteres
        movwf   ADCON1          ;analog RA0 og digital RA1-RA5
```

Listing 6.1: A/D konfiguration i bank1.

Assembler koden listning 6.1 viser den måde, hvorpå PICen er sat op i bank1. Først sættes PORTA til input. Derefter lægges tallet b'00001110' ind i ADCON1 registret. Dette gør, at resultat registrene ADRESH/ADRESL bliver venstre justeret, at ben RA0 bliver analog og RA1-RA5 digitale. Her vælges reference spændingen V_{REF} indirekte til at være V_{DD}, da der ikke vælges et ben, der skal fungere som V_{REF}.
Der skiftes til bank0, hvor b'10000001' lægges ind i ADCON0 (listing 6.2).

```
        bcf     STATUS,RP0      ;Reg.Bank0
        movlw   0081H           ;T_AD = 32T_osc, vælg analog
        mowf   ADCON0           ;port(RA0),converteren tændes
```

Listing 6.2: A/D konfiguration i bank0.

Her vælges T_{AD} til 32 T_{osc}. Dette er den eneste legitime T_{AD} for et 20 [MHz] modul, da de andre værdier for T_{AD} vil være for hurtige (se afsnit B.2.1). I ADCON0 sættes RA0 til det ben, der skal konverteres fra, og ADCen tændes.
Der er i denne opsætning valgt ikke at køre med interrupts.

6.1.2 Programkode til A/D converteren

Når A/D converteren er blevet initialiseret, kan den egentlige programkode skrives (se listing 6.3).

```
ADCSTART:       bsf     ADCON0,2        ;Konverteringen startes
                nop
ADCTEST:        btsfc   ADCON0,2        ;Er ADC færdig?
                goto    ADCTEST
                movf    ADRESH,W
                movwf   ADCDATA
                return
```

Listing 6.3: Programkode for A/D converteren.

Bliver subrutinen ADCSTART kaldt, sættes ADCON0,2 højt. Dette er $GO/\overline{DONE}$ bitten, som sætter A/D konverteringen i gang. Når konverteringen er færdig, bliver denne bit au-

tomatisk sat lav. Det er vigtigt, at converteren får tid til at lade kapacitorstigen op, inden konverteringen sættes i gang. Denne tid kaldes acquisition tiden T_{ACQ} og er typisk 40 [μs] [Microchip-16F87X, 2001]. En T_{ACQ} på 40 [μs] er ikke noget problem at opretholde, da main-løkken venter 90 [ms], og der endvidere skal sendes data mellem master og slave.
Programkoden går derefter i en løkke, hvor den tester på $GO/\overline{DONE}$, for at registrere når konverteringen er færdig. Resultat bliver derefter hentet i ADRESH resultatregistret, hvor de 8 MSb ligger. Dermed kommer de 2 LSb ikke med.
ADRESH bliver lagt ind i ADCDATA, så resultatet er klar til at blive sendt til master.

6.2 Alarm

Alarmsensoren består af en simpel knap, der aktiveret trækker et ben lavt på MCU'en i bilen. Knappens funktion består således i at simulere en alarmsensors input til delsystemet, som omtalt i problemafgrænsningen. Alarmen er kun aktiveret, når bilen er låst.

I hovedprogrammet kaldes rutinen KNAP1_POLL, der tester alarmbenets tilstand. Såfremt benet er lavt lægges en opkode i TX1, mens TX0 er tom. Disse filer Hammingkodes, hvorefter der gives en transmit request, TX_REQ i TX_RDC kontrolregistret.

Et implementeret alarmsystem vil sandsynligvis indeholde flere sensorer forskellige steder i bilen. Idet datafilen TX0 ikke benyttes under transmission af alarm opcode, kan TX0 benyttes til at angive, hvilken sensor i bilen der har udløst alarmen.

6.3 Temperatur

Til dataopsamling af en aktuel kabinetemperatur, vil det være oplagt at anvende en temperaturføler med mulighed for direkte tilslutning til den anvendte microcontroller. Det vil med andre ord sige, at der ønskes en selvstændig kreds med et digitalt output, som indeholder information om en given temperatur. Med anvendelse af en sådan kreds opnås en relativ præcis bestemmelse af den aktuelle temperatur i kabinen.

Som ekstern temperaturføler, der opfylder, de ønskede egenskaber, er der valgt et DS1621 termometer fra Dallas. DS1621 er en selvstændig kreds, der foretager en temperaturkonvertering, når en specifik kommando sendes til kredsen (se appendix E). Den aktuelle temperatur kan herefter aflæses binært fra kredsens EEPROM. Dallas DS1621 kommunikerer via 2 databen, og understøtter kommunikationen med I^2C bus [1]. En nærmere beskrivelse af protokollen og dens opbygning findes i appendix D.

6.3.1 MSSP modulet

Med PIC16F877 som master, varetager denne kontrollen med kommunikationen på bussen. Det vil derfor være oplagt at anvende det såkaldte Master Synchronous Serial Port

[1] Inter-Integrated Circuit

(MSSP)-modul i PIC16F877. MSSP modulet indeholder en I^2C bus controller, og er derfor hardwaremæssig i stand til styre timingen på bussen efter I^2C protokollen. Brugerens opgave er at bestemme, hvilke sekvenser modulet skal udføre, konfigurere modulet til at sende eller modtage data og afgøre hvilke data, der skal sendes på bussen. MSSP modulet er i stand til at starte et interrupt, hvorfor der sættes et SSPIF interruptflag, når hardwaren afslutter en sekvens. I det aktuelle projekt vælges der ikke at anvende interruptfunktionen.

Under anvendelse af MSSP modulet bruges følgende registre af brugeren til konfigurering og styring af I^2C bussen:

SSPCON - Sync Serial Port Control Register.

SSPCON2 - Sync Serial Port Control Register2.

SSPSTAT - Sync Serial Port Status Register.

SSPADD - Sync Serial Port Address Register.

SSPBUF - Sync Serial Port buffer.

I det følgende vil de vigtigste elementer i de enkelte registre blive gennemgået. Først mht. initialisering, og senere mht. styring af selve kommunikationen. Det specifikke kommunikationsflow for temeraturaflæsningen findes i appendiks E.

Initialisering af MSSP modul. Til initialisering af MSSP modulet anvendes fortrinsvis SSPCON Registret. Her kan serielporten aktiveres ved at sætte bitten SSPEN, hvormed de fysiske ben RC3 og RC4 konfigureres til clock og data linie. Inden serielporten aktiveres, er det vigtigt, at benene konfigureres til input i portens register. I SSPCON registreret er der endvidere mulighed for at vælge hvilken I^2C mode, der ønskes. Med de 3 bits SSPCON<3:0> vælges, om der ønskes master eller slave mode og 7 eller 10 bit mode. Opsætning af MSSP modulet til master mode fås ved at sætte SSPCON<3:0> til 1000.

Med ønske om at bruge PICen som masterenhed samt at operere med en kommunikationshastighed på 100 [kHz], vil det være nødvendigt at konfigurere opsætningen af Baud Rate Generatoren (BRG) til 100 [kHz]. I mastermode anvendes SSPADD registeret til at angive BRG frekvensen. Værdien af SSPADD kan for den ønskede clock frekvens findes ud fra ligning 6.2:

$$f_{clock} = \frac{f_{osc}}{4 \cdot (SSPADD + 1)} \tag{6.1}$$

$$\Updownarrow$$

$$SSPADD = \frac{f_{osc}}{4 \cdot f_{clock}} - 1 \tag{6.2}$$

Med en oscillator- og clock frekvens på henholdsvis 20 [MHz] og 100 [kHz], fås ved indsættelse i ligning 6.2 en værdi på 49 = 00110001b.

Den specifikke initialisering af MSSP modulet fremgår af nedenstående listing 6.4:

```
                bcf     STATUS,RP0      ;Skifter til RAM bank0
                movlw   b'00101000'
                movwf   SSPCON          ;Initialiserer SSPCONFIG1

                bsf     STATUS,RP0      ;Skifter til RAM bank1
                movlw   b'00011000'
                movwf   PORTC           ;Initialiserer I/O for PORTC
                movlw   b'10000000'
                movwf   SSPSTAT         ;Initialiserer sspstatus
                movlw   b'00110001'
                movwf   SSPADD          ;Initialiserer BRG til 100 kHz
                movlw   b'01100000'
                movwf   SSPCON2         ;Initialiserer SSPCONFIG2
```

Listing 6.4: Initialisering af MSSP modul

Styring af I^2C **kommunikationen.** Med MSSP modulets I^2C buscontroller administreres kommunikationen gennem registret SSPCON2. Det er brugerens opgave at bestemme de enkelte sekvenser, som buscontrolleren skal udføre. En sekvens påbegyndes ved at sætte det sekvensrelaterede bit i SSPCON2 regiseret. Herefter vil buscontrolleren aktivere baud rate generatoren, og udfører den aktuelle sekvens på datalinien SDA. Når sekvensen er udført, vil hardwaren sætte SSPIF, slette det satte bit i SSPCON2 og standse BRG.

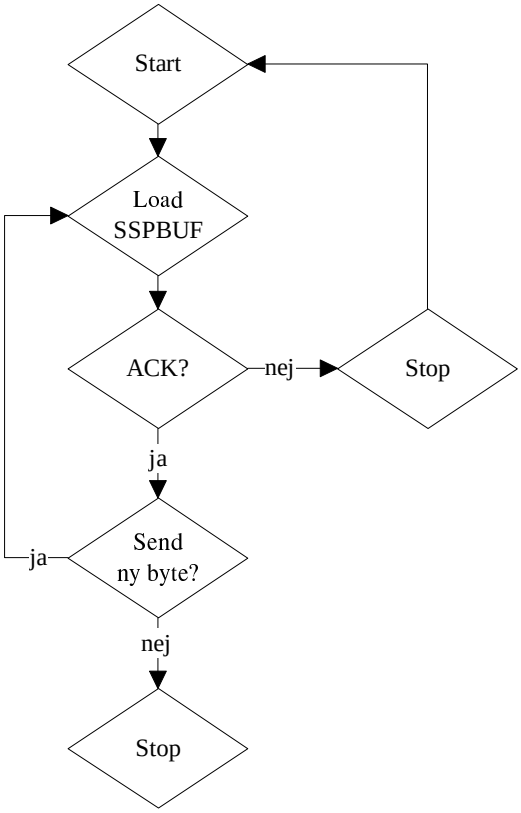
Da buscontrolleren sætter SSPIF flaget efter hver sekvens, er det muligt at påbegynde den næste sekvens umiddelbart efter ved at teste på SSPIF, indtil flaget bliver sat. En startsekvens foretages under denne fremgangsmåde som vist i listing 6.5:

```
STARTCON        bsf     STATUS,RP0      ;Skifter til RAM bank1
                bsf     SSPCON2,SEN     ;Generere en start sekvens
                bcf     STATUS,RP0      ;Skifter til RAM bank0
                btfsc   PIR1,SSPIF      ;Er startsekvens gennemført?
                goto    $-1
                return
```

Listing 6.5: Startsekvens.

Når en startsekvens er udført, kan data sendes en byte af gangen ved blot at læse den specifikke databyte ind i bufferen SSPBUF. Buscontrolleren aktiverer igen baud rate generatoren, og begynder at latche data på SDA, indtil alle 8 bit er afsendt. Efter den 9. clock cycle sættes SSPIF flaget, og status på acknowledge (DS1621) læses ind i ACKSTAT. Er statusbitten

høj, sendes en stopsekvens, og transmitteringen gentages. En typisk sendestruktur kunne forløbe som vist på figur 6.1.



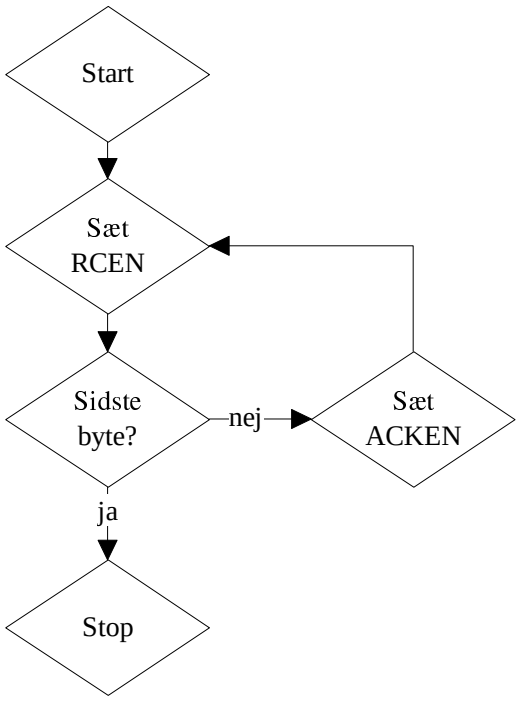
Figur 6.1: Flow over I^2C sendeprocedure.

Når data skal modtages fra DS1621, sættes RCEN bitten. Når den 8. bit er roteret ind i SSPBUF, skal PICen returnere status på acknowledge. Hvis flere bytes skal modtages, sættes ACKEN bitten i SSPCON2 registeret mellem hver byte. Buscontrolleren returnerer derved indholdet af ACKDT på den 9. clock cycle. ACKDT cleared angiver et acknowledge, mens ACKDT sat angiver et not acknowledge. Når sidste byte er modtaget, termineres transmissionen ved at sende en stopsekvens. Da SDA releases efter den 8. bit, modtager DS1621 et not acknowledge, idet pull up modstanden trækker SDA høj under den 9. clock cycle.

Receivemode aktiveres jf. listing 6.6, og flowstrukturen kan forløbe som vist i figur 6.2.

```
RECEIVE         bsf     STATUS,RP0      ;Skifter til RAM bank1
                bsf     SSPCON2,RCEN    ;Læser data ind i SSPBUF
                bcf     STATUS,RP0      ;Skifter til RAM bank0
                btfsc   PIR1,SSPIF      ;Er alle 8 bit modtaget?
                goto    $-1
                return
```

Listing 6.6: Modtagerutine for MSSP modulet.



Figur 6.2: Flow over I^2C modtagerprocedure.

En stopsekvens foretages på tilsvarende måde som start- og receiveprocedurerne ved at sætte PEN som vist i listing 6.7

```
STOPCON         bsf     STATUS,RP0      ;Skifter til RAM bank1
                bsf     SSPCON2,PEN     ;Generere en start sekvens
                bcf     STATUS,RP0      ;Skifter til RAM bank0
                btfsc   PIR1,SSPIF      ;Er stopsekvens gennemført?
                goto    $-1
                return
```

Listing 6.7: Stopsekvens.

Med de enkelte sekvensprocedurer administreres I^2C kommunikationen på bussen mellem PICen og DS1621. Procedurene indgår som subrutiner og kaldes af mastermodulet. Strukturen over kommunikationen findes i appendix E.
Data, som indlæses i SSPBUF, indeholder udelukkende information om temperaturen. I projektet vælges at aflæse temperaturen med en opløsning på 1 [°C]. Når den 8. bit er indlæst i SSPBUF, flyttes indholdet til et register, som aflæses under forespørgsel på temperaturen. [Microchip-Midrange, 1997]

6.4 Delkonklusion

Som nævnt i kravsspecifikationen er sensorer og aktuatorer ikke implementeret i sin fulde udstrækning. I stedet er der lagt vægt på kommunikationen mellem MCUen og sensorerne samt aktuatorerne.

Der er følgelig udviklet softwaremoduler til aflæsning af benzinstand, temperatur og alarm. Herunder benyttes A/D converteren og I^2C bussen. Herigennem er princippet i kommunikationen mellem delsystemet og det fysiske system udgjort af bilen vist. Delsystemet opfylder følgelig de opstillede krav til dataopsamling i henhold til kravsspecifikationen.

7 Test af konstruktionen

Formålet med dette kapitel er at opstille en række tests af konstruktionen, udfra hvilke det kan vurderes, om dele af kravsspecifikationen er opfyldt.

7.1 Forsøgsserier

Funktionaliteten er afgørende for, at systemet virker efter hensigten og reagerer korrekt på brugerens input. Dette kræver gennemtestede softwaremoduler. Alle softwaremoduler er testet og debugget i MPLAB. Efterfølgende er systemets software konstateret som værende stabilt og fuldt funktionelt.

For at dokumentere funktionaliteten af systemet foretages en testrække. En overholdelse af rækkevidden af radiomodulerne og en nøjagtig A/D omsætning er nødvendig for systemets funktionalitet. Sekundært vil et minimeret effektforbrug også være af interesse. Der er foretaget følgende forsøg med systemet:

- Rækkevidde.
- Nøjagtigheden for A/D converteren.
- Effektforbrug.

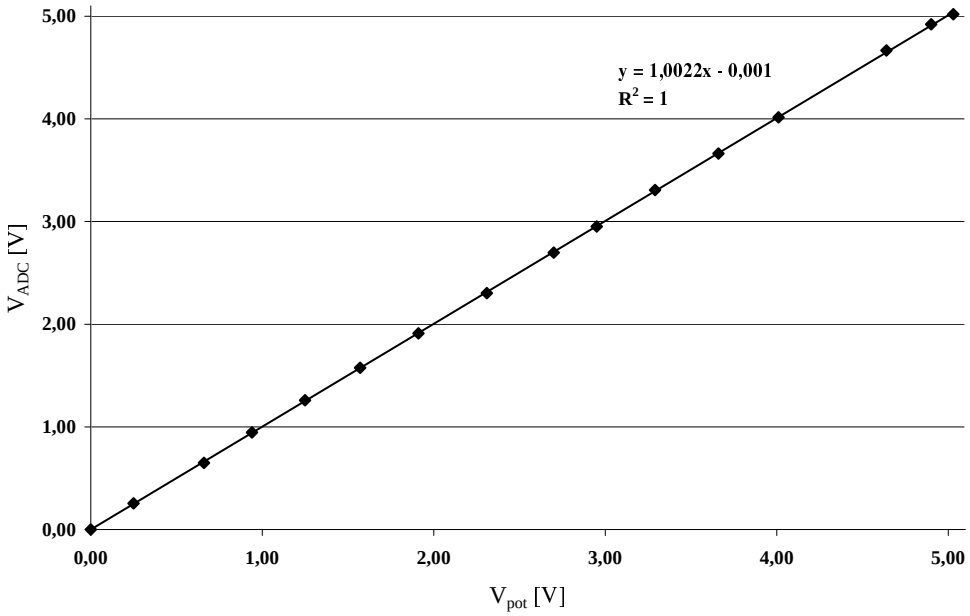
7.2 Rækkevidde

Rækkevidden er målt til 57 [m], hvilket er mere end forventet. Denne rækkevidde kan øges ved at ændre på antennedesignet eller benytte andre radiomoduler med en højere sendeeffekt eller følsomhed. For dette projekt er rækkevidden tilstrækkelig, idet den opfylder kravsspecifikationen.

7.3 A/D converteren

For at undersøge nøjagtigheden af A/D converteren er der foretaget en forsøgsserie, hvor spændingen over et potentiometer måles med et voltmeter. Ved at måle referencespændin-

gen V_{REF}=5.03 [V] for systemet kan de 8 mest betydende cifre i A/D converterens dataregistre ved udlæsning omsættes til spændingen over potentiometret målt af A/D converteren. Denne spænding afbildes som funktion af spændingen målt med voltmeter. Målepunkterne skal således ligge på en ret linie gennem origo med en hældningskoefficient på 1 [V/V].



Figur 7.1: Output V_{ADC} fra ADC som funktion af spændingen V_{pot} over potentiometret. Referencespændingen er V_{REF}=5.03 [V]

Det fremgår af grafen figur 7.1, at målepunkterne tilnærmelsesvist ligger på en ret linie. Der kan være små usikkerheder i forbindelse med måleudstyret og A/D converterens 10 bit opløsning. Der er således ikke grund til at betvivle A/D converterens præcision.

7.4 Temperatur

For at kontrollere pålideligheden af DS1621, er der målt flere temperaturer, som er sammenholdt med to andre digitale termometre. I alle tilfælde var der samstemmelse mellem måledata. Det må dog bemærkes at DS1621 er relativt langt tid om at måle en stabil rumtemperatur, hvis den udsættes for pludselige temperaturændringer. Dette giver ikke problemer, idet termometeret tænkes fast monteret i en bil.

	P16F84A			P16F877		
	V [V]	I [mA]	P [mW]	V [V]	I [mA]	P [mW]
Standby	5.01	24	$1.2 \cdot 10^2$	5.02	25	$1.3 \cdot 10^2$
Peak	5	30	$2 \cdot 10^2$	5	35	$2 \cdot 10^2$

Tabel 7.1: Effektforbrug for delsystemer. Ved peak værdier er spænding målt med et betydende ciffer.

7.5 Effektforbrug

Effektforbruget for henholdsvis mastersystemetfjernbetjening og bilen er målt ved stand by samt ved peak, hvor delsystemet kræver den største effekt. Dette er for begge delsystemer under transmission. Effekten er målt for systemet eksklusiv effektforbruget i LM78L05.

Såfremt prototypen udvikles yderligere, vil en nedbringelse af effektforbruget have stor prioritet. Det nuværende standby effektforbrug er meget højt med en kort batterilevetid i fjernbetjeningen til følge. Effekten kan nedbringes ved at sætte clock frekvensen for de to MCUer ned samt indføre sleep styring af de to MCU. LM78L05, der er benyttet til spændingsregulering, kan evt. udelades i et færdigt design, hvis der benyttes en anden batteritype end det nuværende batteri på 9 [V].

7.6 Delkonklusion

Forsøgsrækken har vist, at kommunikationen mellem delsystemerne og A/D converteren virker tilfredsstillende og lever op til kravsspecifikationen. Resten af systemet er testet ved brug og udviser stabilitet og den ønskede funktionalitet. Systemet bestående af prototypen er i kørende stand og erklæres fuld funktionsdygtigt.

8 Konklusion

8.1 Opsummering

Igennem dette projekt er der opbygget et system, bestående af to PIC microcontrollere, som er i stand til at kommunikere bidirektionalt. Formålet var at skabe kommunikation mellem en bil og en fjernbetjening til brug i forbindelse med en tyverialarm. I projektet er der lagt vægt på den tekniske løsning og ikke produktparametre som f.eks udformning og produktets integration. Projektet skal derfor ses som en prototype, der kan videreudvikles til et færdigt produkt, der kan integreres i en bil.

Ud fra problemformuleringen er der opbygget to microprocessorsystemer, der opfylder kravsspecifikationen. Det samlede system er i stand til at kommunikere bidirekttionalt igennem en til formålet udviklet protokol. Der foretages fejlkontrolkodning af de data, der overføres mellem delsystemerne, og der kan opsamles data fra udvalgte dele af omverdenen. Sluttelig er der udviklet et brugerinterface til simpel interaktion mellem bruger og systemet.

For at udvikle det samlede system er dette nedbrudt i delsystemer, og grænseflader mellem disse er defineret. Dette giver mulighed for senere udvidelser og optimeringer af de enkelte softwaremoduler i systemet. Efterfølgende er der foretaget en syntese af delsystemernes software- og hardwaremoduler. For at dokumentere funktionalitet og stabilitet af systemet er der foretaget en testrække. Systemet har vist sig at være stabilt og opfylder den ønskede funktionalitet. Ydermere giver inddelingen i moduler og delsystemer stor mulighed for at foretage udvidelser på systemet.

8.1.1 Muligheder/udvidelser

Som nævnt i problemanalysen er der mange muligheder for at udvide systemet til at omfatte personlige indstillinger i bilen, alt efter hvilken fjernbetjening der benyttes. PICen i bilen kan således styre aktuatorer og modtage input fra mange andre sensorer end de eksisterende. Der er tilstrækkelig hukommelse og porte til disse udvidelser af delsystemet i bilen.

Endvidere kan brugerfladens kompleksitet øges ved at opbygge et menusystem på fjernbetjeningen, så flere informationer kan formidles på en overskuelig måde, ligesom der kan vælges et grafisk display med mulighed for symbolsk notation i displayet.

Størrelsen af systemet bør også gøres mindre ved en fuld implementering. Især fjernbetjeningen med display fylder meget, og det vil ganske givet være et salgsargument, at systemet ikke fylder væsentligt mere end de nuværende bilnøgler og nøgleringe.

Protokollen indeholder ligeledes et potentiale for udvidelse med en kryptering af de data, der transmitteres. Dette giver en øget beskyttelse mod hacking, hvor bilen oplåses af personer med det rigtige udstyr og de forkerte hensigter.

8.1.2 Begrænsninger

Hukommelsespladsen i MCUen i fjernbetjeningen er begrænset. Der er allerede foretaget nogle optimeringer af programkoden, men en udvidelse af protokollen med kryptering vil kræve en yderligere optimering, en større PIC MCU eller en PIC, der er dedikeret til kryptering og dekryptering af data. Dette kan, alt efter mængden af data, der skal krypteres, kræve en ændring af transmissionshastigheden.

En evt. mere kompleks brugerflade vil ligeledes kræve ekstra hukommelsesplads.

8.2 Tilegnede færdigheder

Formålet med projektet har været at opnå viden om systembegrebet og forståelse for metoder til konstruktion af digitale systemer herunder programmeludvikling. Dette skal give grundlag for at kunne foretage specifikation, konstruktion, realisation, test og dokumentation af et microprocessorsystem.

Gennem selvstudie og de oversigtsskabende SE og PE kurser i dette semester, er der truffet valg for projektets microcontrollere, for at opnå formålet med projektet og løse den valgte problemstilling samt opfylde kravsspecifikationen.

Inden for microcontrollere er der opnået viden og forståelse inden for områderne:

- Assemblerprogrammering.
- Timing.
- Eksekveringstid.
- Interrupthåndtering.
- Håndtering af registre.
- Adressering af hukommelse.

Analyse af de tilsluttede perifere enheder har givet viden og forståelse inden for områderne:

- Konvertering af analoge signaler til digital via ADC.
- Opsætning og brug af et standard bussystem I^2C.

- Timing af radiomoduler, skifteregister, display og temperaturføler.
- Udvikling af protokol.
- Bidirektional radiokommunikation i et master/slave forhold.
- Fejlkontrolkodning af data.

Idet de opstillede krav til projektet er opnået og de fremstillede delsystemer fungerer, må formålet med projektperioden siges at være nået.

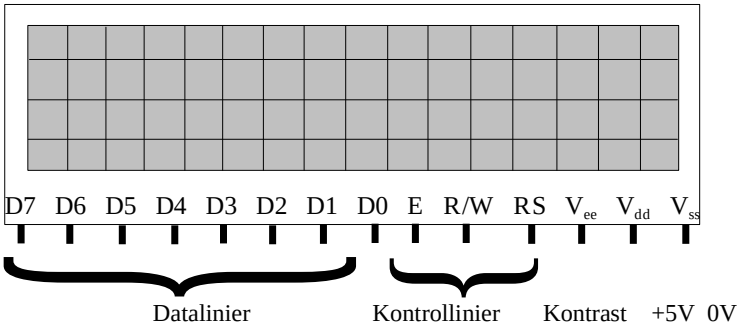
Liquid Crystal Display

A

I projektet benyttes et Liquid Crystal Display (LCD) til kommunikation med brugeren. I følgende afsnit gennemgås displayets opsætning

A.1 Displayets opbygning

Displayet er opbygget af 4 linier med 16 karakterer, hvor hver karakter indeholder 5x7 pixels. Displayet indeholder en Hitachi controller-chip HD44780 samt hukommelse. Displayet er ideelt til brug sammen med PIC-microcontrollere, da indgangene er TTL kompatible. Displayet har en indbygget tegntabel, som opfylder 7-bit ASCII. Hvert tegn fylder dermed én byte, som indlæses via displayets datalinier. Styringen af displayets controller foretages af tre kontrol linier, samt to til spændingsforsyningen. [Seiko-Instruments, 1998]



Figur A.1: Diagram over display.

På fig. A.1 ses otte datalinier samt de tre kontrollinier, der består af R/W (Read/Write), E (Enable) og RS (Register Select). Den elektriske tilstand af R/W afgør, om der skrives til eller læses fra displayet. Udlæsning af data-linierne til displayet foretages kun, når E er trukket høj. Med RS kan der skiftes mellem displayet to modes, hhv. character- og commandmode:

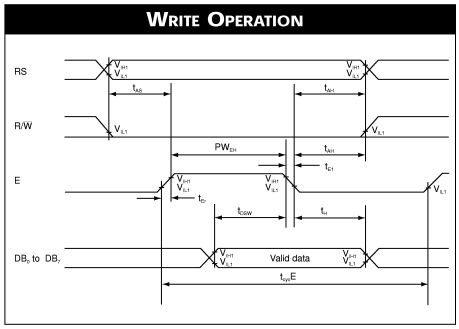
Character: Matcher den indlæste byte med tabelopslag og udskriver tegnet.

Command: Opsætning af display mht. initialisering, cursor position og form, samt lagring af symboler i hukommelsen.

Kontrasten kan justeres ved at trække V_{ee} mellem forsyningsspænding og 0[V].

A.2 Timing

For at undgå data overflow og sikre datastabilitet, således at displayet kan følge med, er det nødvendigt at time data og kontrollinierne. Under udlæsning af data på datalinierne skal E sættes lav, i minimum 230 [ns]. Når data er klar, trækkes E høj. Positionen skal holdes i min. 230 [ns] før E atter kan skifte til lav. Der er også minimums tider for timing af RS og datalinierne, hvilket fremgår af timing diagrammet på figur A.2. Endvidere må flankerne maksimalt tage 20 [ns]. Timing af displayet er ikke kritisk, så længe minimunstiderne overholdes. Tilgangstiden til displayet kan være længere, hvilket betyder, at PICen uden problemer kan tilpasses, da hver af PICens instruktioner tager mindst 200 [ns].[Seiko-Instruments, 1998]



(a) Timing diagram

WRITE TIMING CHARACTERISTICS

V_{DD}=5.0V±5%, V_{SS}=OV, T_A=0°C TO 50°C

Item	Symbol	Standard		Unit
		Min.	Max.	
Enable cycle time	$t_{CYC}E$	500	—	ns
Enable pulse width High Level	PW_{EH}	230	—	ns
Enable rise and fall time	t_{ER}, t_{EF}	—	20	ns
Address setup time RS,R/W—E	t_{AS}	40	—	ns
Address hold time	t_{AH}	10	—	ns
Data setup time	t_{DSW}	80	—	ns
Data hold time	t_H	10	—	ns

(b) Tabel over tider

Figur A.2: Timing af LCD modul.

A.3 Initialisering

Efter power-on, er det nødvendigt at initialisere displayet, før displayet kan udskrive tegn. Initialiseringen foretages ved at sende bestemte kommandoer til opsætning af displayet. Under initialiseringen skal der anvendes længerevarende delays mellem de afsendte kommandoer. Dette er for, at displayets egen controller kan nå at blive klar. Kommandoerne er:

Reset sequence: 03h, 03h, 03h, 02h

Function set: 02h, 08h: 4bit data length, 2 linie, 5x7 dot.

Entry mode set: 0h, 06h: increment, no shift.

Display on/off control: 0h, 0Eh: display on, cursor on, blink off.

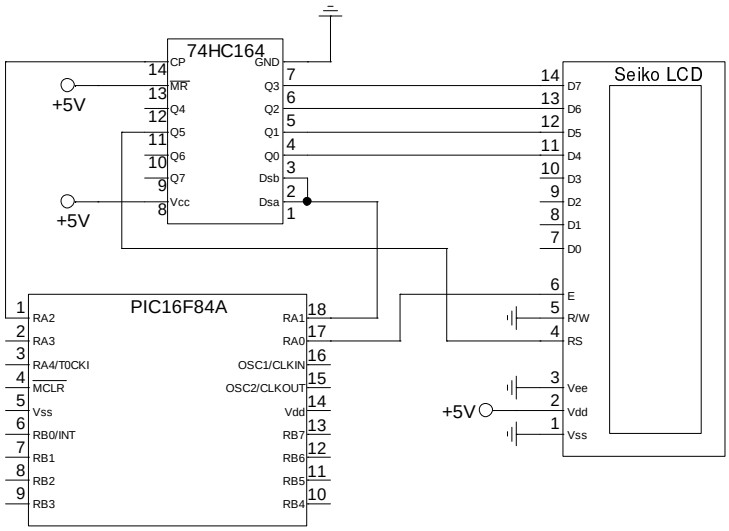
Display clear: 00h, 01h

Efterfølgende, kan der, sendes kommandoer om indlæsning af symboler til displayets hukommelse. Alle kommandoer under initialiseringen sendes i command-mode. Inden initialiseringen starter, skal der afvikles et delay på minimum 15 [ms]. Delayet mellem kommandoerne varierer fra 4.1 [ms] til 1 [μs], hvorfor der vælges et fast delay på 5 [ms]. Initaliserningen slutter med et delay på 2 [ms]. Ialt tager initialiseringsekvensen 77 [ms] inden displayet er klar til at modtage tegn.[Seiko-Instruments, 1998]

A.4 Fysiske forbindelser

Displayet kræver forbindelse til PICen via E, RS og 8 eller 4 datalinier. I opstillingen benyttes 4 datalinier, hvilket er specificeret under initialiseringen.
R/W og kontrast pin hardwires, da disse i denne opstilling ikke skal ændre tilstand.
Displayet optager for mange I/O pin på PICen, hvorfor der benyttes et serielt til parallel 8bit skifteregister (74HC164), for at begrænse de benyttede I/O pin til blot 3 i stedet for 6.
E linien kobles direkte til PICen, mens RS og data linierne latches serielt ind i skifteregisteret via én I/O pin. For hvert bit, der latches ind, bruges en clockpuls. Skifteregisterets

clockpuls styres af PICen. Endvidere kobles skifteregisterets $\overline{MR}$ (Master Reset) til forsyningsspænding. På fig. A.3 ses opstillingen.



Figur A.3: LCD opstilling

På figur A.3 ses, at clocken er tilsluttet PICen. Således kan clocken kontrolleres via PICens software, for at få timingen til at passe.

A.5 Software

Til styring af displayet indlæses software i PICen. Softwaren er opdelt i subrutiner, der hver varetager bestemte funktioner. Den vigtigste rutine er udlæsningsrutinen til skifteregisteret. I rutinen roteres indholdet af en 8 bit hukommelsescelle til serielt output på én af PICens I/O pins.

Nedenstående programkode listing A.1 viser hvorledes ét bit udlæses til skifteregisteret efterfulgt af en clockpuls.

```
udlas       bcf     PORTA,E         ;sætter E lav
            movlw   08h             ;starter tæller
lcd_loop    movwf   lcd_count       ;flytter tællerværdi til lcd_count
            bcf     PORTA,dat       ;sætter data I/O til 0
            rlf     lcd_dataud      ;flytter næste bit til carry
            btfsc   STATUS,C        ;tester om carry er 0
            bsf     PORTA,dat       ;sætter data I/0 til 1, hvis carry 0
            bsf     PORTA,clock     ;sætter skifteregister_clock høj
            bcf     PORTA,clock     ;sætter skifteregister_clock lav
            decfsz  lcd_count,f     ;loop-rutine for udlæsning
            goto    lcd_loop        ;af alle 8 bit
            bsf     PORTA,E         ;E sættes høj
```

Listing A.1: Struktur for udlæsning til display.

Rutinen gentages otte gange for at få udlæst hele hukommelsescellen, der i koden hedder lcd_dataud. Inden udlæsningen starter, sættes E linien lav. Efter skifteregisteret har modtaget 8 bit, sættes E atter høj. Herved bliver displayet ikke påvirket, når der udlæses data til skifteregisteret.

Da der kun er én hukommelsescelle til udlæsning, skal alt, hvad der ønskes udlæst til displayet, først lægges ind i denne celle, hvorefter et kald af udlæsningsrutinen er nødvendigt. Udlæsningsrutinen kan kaldes af andre rutiner såsom initialisering, menusystemet, bcd m.m. hvor disse rutiner kan styre displayet.

Der benyttes en 4 bit tilgang til displayet, så alle kommandoer og karakterer splittes op i to pakker, der udskrives hver for sig. RS skal defineres for hver pakke. Følgende runtine angiver hvorledes lcd_dataud opsplittes:

```
lcd_rs_h    bsf     _rsset,3        ;Sætter RS høj (character-mode)
lcd_rs_l    movf    lcd_dataud,w    ;flytter uddata til W
                                    ;(RS er lav)
            movwf   f_nybb          ;gemmer uddata
            call    lcd_cmd_set     ;kalder command-bit rutine
                                    ;til last_nybble
            movwf   l_nybb          ;flytter W til last_nybble
            swapf   f_nybb,w        ;swap nybble i first_nybble
            call    lcd_cmd_set     ;kalder command-bit rutine
                                    ;til first_nybble
            movwf   lcd_dataud      ;flytter W (first_nybble)
                                    ;til lcd_dataud
            call    udlas           ;udskriver første nybble
            movf    l_nybb,w        ;flytter last nybble til W
            movwf   lcd_dataud      ;flytter W til lcd_dataud
            call    udlas           ;udskriver anden nybble
            clrf    _rsset          ;renser _rsset temp RAM
            return
```

Listing A.2: Opsplitning af lcd_dataud.

Kald af lcd_rs_h eller lcd_rs_l angiver, om RS skal sættes høj eller lav. Subrutinen i listing A.3 sætter bitten for tilstanden af RS til hukommelsescellen, der udlæses til skifteregisteret.

```
lcd_cmd_set movwf   lcd_temp        ;flytter W til lcd_temp
            bcf     lcd_temp,4      ;clr Q4 på skifteregister
            btfsc   _rsset,3        ;tester om RS skal være høj
            bsf     lcd_temp,5      ;sætter RS høj
            btfss   _rsset,3        ;tester om RS skal være lav
            bcf     lcd_temp,5      ;sætter RS lav
            bcf     lcd_temp,6      ;clr Q6 på skifteregister
            bcf     lcd_temp,7      ;clr Q7 på skifteregister
            return
```

Listing A.3: Indsættelse af RS.

A/D converter

B

For at behandle måledata fra analoge sensorer, der er tilkoblet PIC16F877, er det nødvendigt med en analog til digital konvertering. En sådan converter er integreret i PIC16F877. Dette kapitel giver et indblik i dens funktionalitet og design.

B.1 Konstruktion

Den implementerede A/D converter kan have op til 8 analoge input alt efter konfiguration, som resulterer i en konvertering til en 10 bit digital værdi. Konverteringen sker ved successiv approksimation ved hjælp af en kapacitorstige, der switcher en reference spænding V_{REF}. Dette skaber en spændingsændring til en analog comparator, der fungerer som en balance vægt, for hvilken værdi det digitale resultat har. Denne metode kaldes også for redistribution metoden.

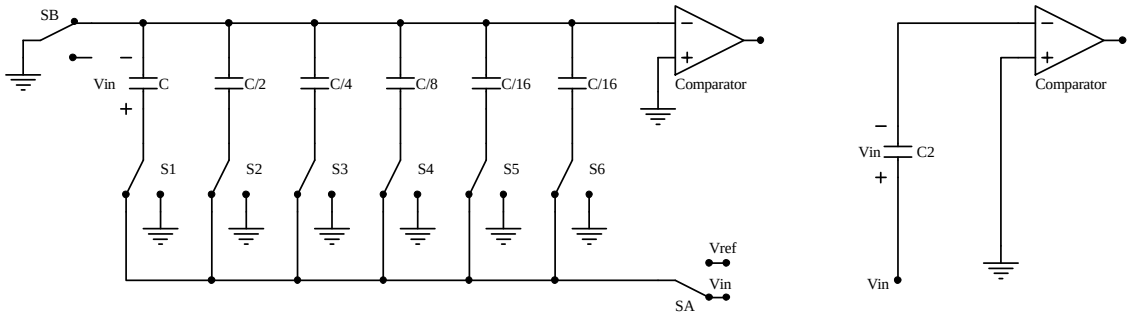
A/D omsætningen foregår i 3 stadier. Et **sample** stadie hvor kapacitorerne oplades, et **hold** hvor comparatoren initialiseres og et **redistributions** stadie, hvor den egentlige konvertering sker.

A/D princippet vil blive illustreret med et forenklet 5 bit converter eksempel.

B.1.1 Sample stadiet

Switch SB er lukket, og switch SA er koblet til V_{in}, som lader kapacitorerne op (Se figur B.1). Kapacitorerne bliver ladet op med en total ladning $Q_{in} = 2C \cdot V_{in}$ summen af den samlede kapacitans multipliceret med den analoge input spænding.

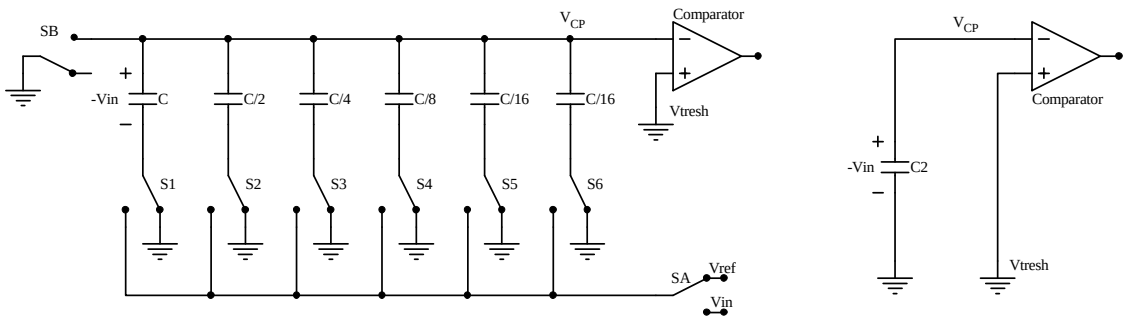
Det sidste led i kapacitorstigen $C/16$ er tilstede, for at summen af kapacitanserne er $C_{total} = 2C$. [Kugelstadt, 2000]



Figur B.1: Sample stadiet for en 5 bit A/D converter.

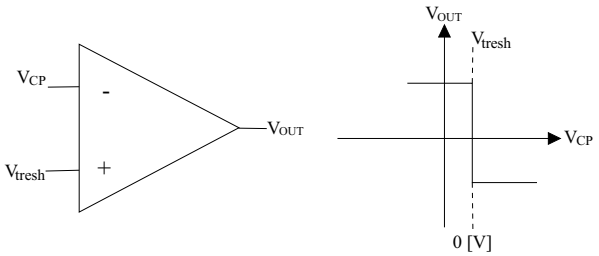
B.1.2 Hold stadiet

Når kapacitorerne er opladet, åbnes SB, SA switcher til V_{REF} og switchene S1-S6 sættes til jord. Dermed bliver spændingen over kapacitorerne $-V_{in}$. Dette skaber spændings potentialet V_{CP} ved comparatorens indgang, hvor $V_{CP} = -V_{in}$. (Se figur B.2).



Figur B.2: Hold stadiet for en 5 bit A/D converter.

Spændings potentialerne V_{CP} og V_{tresh}, der er tilkoblet jord, bruger comparatoren til at bestemme, om V_{OUT} skal være logisk høj eller lav (se figur B.3).



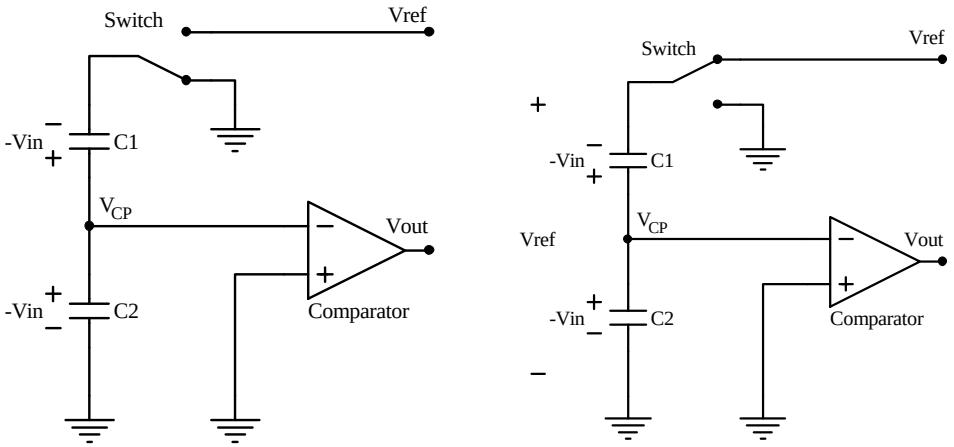
Figur B.3: Diagram over comparatoren.

Dvs., er $V_{CP} > 0$, bliver comparatorens udgang lav for det pågældende bit. Er $V_{CP} < 0$ bliver udgangen høj. [Kugelstadt, 2000]

B.1.3 Redistributions stadiet

Her begynder switchene S1-S5 konsekutivt at skifte, for ved hjælp af successiv approksimation at finde en tilnærmet værdi for V_{in}. Når en switch skifter bliver den pågældende kapacitor koblet til V_{REF}. Dermed går kapacitoren fra at sidde i parallel med de andre til at sidde i serie (se figur B.4). Switchene skifter altså, så kapacitorerne blive delt op i 2 grupper, hvor C_1 er de kapacitorer, hvis switch er tilsluttet V_{REF}, mens C_2 er kapacitorerne forbundet

til jord. En kapacitor, der bliver tilslutte V_{REF}, vil skabe en spændingsændring af potentialet V_{CP}, hvor ændringen af V_{CP} er en spændingsdeling mellem C_1 og C_2.



(a) Kapacitorerne C_1 og C_2 sidder parallelt.

(b) Kapacitor C_1 sidder i serie med C_2

Figur B.4: Ækvivalent diagram over kapacitorstigen før og efter en tilkobling til V_{REF}.

Spændingsdeling mellem kapacitorer. For at bestemme spændingsdelingen mellem to kapacitorer benyttes KVL, samt at kapacitorer i serie bliver opladet med samme ladning Q. Først udledes V_2 udfra at $Q_1 = Q_2$,

$$Q_1 = Q_2 \tag{B.1}$$

$$\Updownarrow$$

$$C_1 \cdot V_1 = C_2 \cdot V_2 \tag{B.2}$$

$$\Updownarrow$$

$$V_2 = \frac{C_1}{C_2} \cdot V_1 \tag{B.3}$$

Derefter indsættes ligning B.3 i KVL.

$$V_{REF} = V_1 + V_2 \tag{B.4}$$

$$\Updownarrow$$

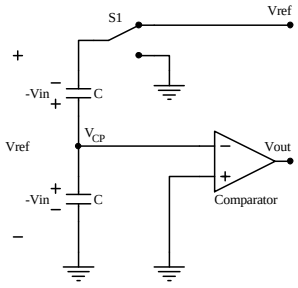
$$V_{REF} = V_1 + \frac{C_1}{C_2} \cdot V_1 \tag{B.5}$$

$$\Updownarrow$$

$$V_1 = \frac{C_2}{C_1 + C_2} \cdot V_{REF} \tag{B.6}$$

Spændingen V_1 kan nu bestemmes, og dermed kan ændringen af V_{CP} ved de forskellige stadier beregnes.

Stadie 1. Figur B.5 illustrerer det første stadie af konverteringen, hvor kapacitor C sidder i serie med de resterende kapacitorere. Gennemgås stadie 1 skridt for skridt begynder switch S1 med at koble sig til V_{REF}. Til tiden t^-, hvor S1 stadig er koblet til jord er $V_{CP} = -V_{in}$. Ved tiden t^+, hvor S1 er koblet til V_{REF}, er der stadig spændingen $-V_{in}$ over kapacitorerne. Derudover opstår der et spændingsfald V_{REF} over C_1 serielt med C_2, og dermed en ændring af potentialet V_{CP}. Til at bestemme spændingsændringen af V_{CP} bruges ligning B.6, der adderes med spændingen $-V_{in}$, som er opladt i kapacitorerne. Værdien for V_{CP} bliver:



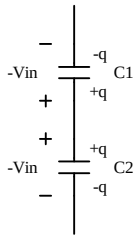
Figur B.5: Stadie 1.

$$V_{CP} = -V_{in} + \frac{C}{C+C} \cdot V_{REF} \;\Rightarrow\; V_{CP} = -V_{in} + \frac{V_{REF}}{2} \tag{B.7}$$

Der er ved første stadie blevet tilført $\frac{1}{2}V_{REF}$ til V_{CP}. Hvis $V_{CP} < 0$ bliver comparatorens udgang logisk høj. Er $V_{CP} > 0$ bliver udgangen logisk lav.

Under stadie 1 har V_{REF} påført en ladningsændring til kapacitorerne, så kapacitorerne hver især ikke længere har den ladning, der blev tilført under sample stadiet. Alligevel viser det sig, at den samlede ladning stadig er $Q_{total} = C_{total} \cdot V_{in}$. Dette fører tilbage til at kapacitorer

i serie bliver ladet op med den samme ladning Q. Da ladningen i C_1 er blevet vendt i forhold til C_2, vil den påførte ladning af C_1 blive negligeabel i C_2.



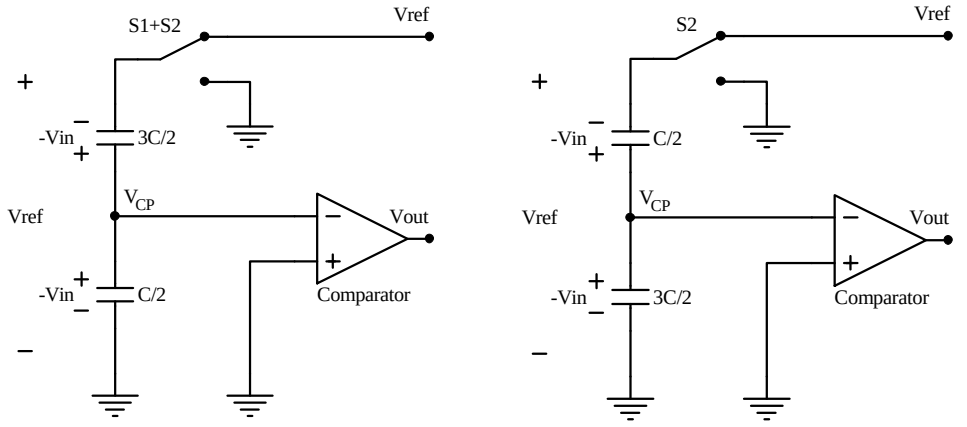
Figur B.6: C_1 og C_2 i serie.

Dette er illustreret på figur B.6. Den ændring der sker på den øverste plade af C_1, vil også ske på den nederste plade af C_2 men med modsat fortegn.
Efter stadie 1 kan der opstå to situationer for stadie 2.

Stadie 2 med lav stadie 1. Er comparatorens output logisk lav efter stadie 1, er $-V_{in} + \frac{1}{2}V_{REF} > 0$. Næste stadie vil derfor være at sænke spændingsfaldet over C_1 ved at forøge dens kapacitans. Dette er illustreret på figur B.7(a). Her er switch S1 stadig koblet til V_{REF}. S2 bliver nu tilsluttet, så den samlede kapacitans for $C_1 = \frac{3}{2}C$. Når $C/2$ bliver koblet til V_{REF}, vendes ladningen i kapacitoren i forhold til V_{REF}. Dermed er noget af den ladning, der blev overført fra C_1 til C_2 under 1. stadie tilbage i C_1. På den måde vil ladningen Q_{total} i kapacitorstigen kunne holdes konstant under hele konverteringen. Da spændingen $-V_{in}$ ikke ændrer sig, kan den analoge input spænding bestemmes ved ændring af spændingsfaldet V_1. Dermed kan ligning B.6 bruges for samtlige stadier af konverteringen.

Det nye potentiale for V_{CP} bliver dermed:

$$V_{CP} = -V_{in} + \frac{\frac{C}{2}}{\frac{3C}{2} + \frac{C}{2}} \cdot V_{REF} \Rightarrow V_{CP} = -V_{in} + \frac{V_{REF}}{4} \tag{B.8}$$



(a) Stadie 2 hvor stadie 1 var lav

(b) Stadie 2 hvor stadie 1 var høj

Figur B.7: Ækvivalent diagram over kapacitorstigens andet stadie.

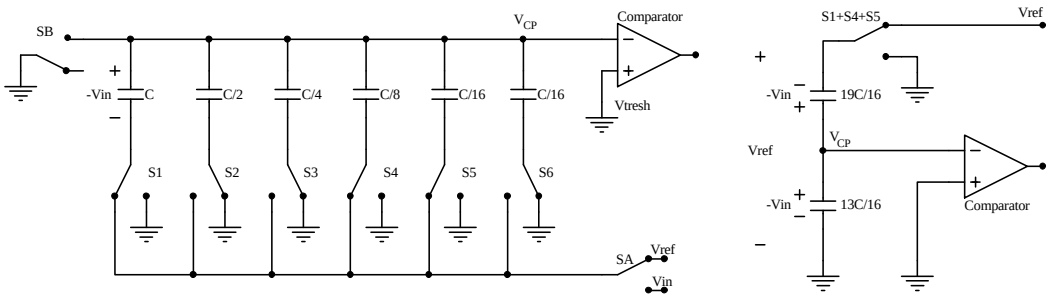
Stadie 2 med høj stadie 1. Er comparatorens output logisk høj efter stadie 1, er $-V_{in} + \frac{1}{2}V_{REF} < 0$. Næste stadie vil derfor være at hæve spændingsfaldet over C_1. Dette er illustreret ved figur B.7(b). Her er switch S1 koblet tilbage til jord, så spændingsfaldet igen er V_{CP} over kapacitorerne. S2 bliver nu tilsluttet V_{REF}, og det nye potentiale beregnes:

$$V_{CP} = -V_{in} + \frac{\frac{3C}{2}}{\frac{3C}{2} + \frac{C}{2}} \cdot V_{REF} \Rightarrow V_{CP} = -V_{in} + \frac{3V_{REF}}{4} \tag{B.9}$$

Dermed bliver der ved andet stadie tilført $-\frac{3}{4}V_{REF}$ til V_{CP}. Konverteringen forsætter nu til stadie 3.

Resterende stadier. Sådan fortsætter samtlige stadier til konverteringen er færdig ved switch S5 og $V_{CP} \simeq 0$, dog vil der kunne være en fejl på $\pm\frac{1}{2}LSb$, da $\frac{\Sigma\, C_{1,total}}{\Sigma\, C_{total}} = \frac{31}{32}$.
Ses der på figur B.8, der forestiller kapacitorstigen lige efter en konvertering, vil den påførte analoge indgangsspænding være:

$$V_{in} = \frac{\frac{13C}{16}}{\frac{19C}{16} + \frac{13C}{16}} \cdot V_{REF} \Rightarrow V_{in} = \frac{13V_{REF}}{32} \tag{B.10}$$



Figur B.8: kapacitorstigen efter en konvertering.

Ved betragtning af switchene S1-S5, kan det ses, at S1, S4 og S5 er koblet til V_{REF}. I dette tilfælde er V_{OUT} på comparatoren lav. Ved switch S3 og S4 der er koblet til jord vil V_{OUT} være sat højt. Dermed vil det digtale resultat have den binære værdi 00110, for $V_{in} = \frac{13}{32} \cdot V_{REF}$. [Kugelstadt, 2000]

B.1.4 Registre

Alt kommunikation med A/D converteren foregår gennem 4 registre:

ADRESH MSB af A/D konverteringen

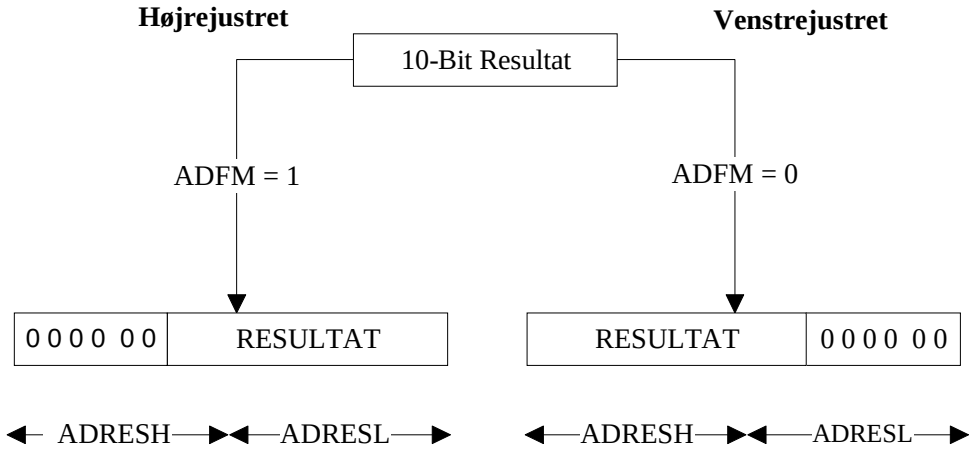
ADRESL MSB af A/D konverteringen

ADCON0 Operativt register til ADC

ADCON1 Opsætning af ADC

De første 2 registre er til oplagring af resultatet fra konverteringen. Til sammen danner de et 16 bit register par, som indeholder et resultat på 10 bit. Dette giver en mulighed for at højre- eller venstrejustere resultatet. Denne funktion styres af bitten ADFM i ADCON1. (Se figur

B.9)



Figur B.9: Lagring i resultat registrene.

De 2 næste er kontrol registre, hvor ADCON0 styrer den operative del af A/D converteren, og ADCON1 er til konfiguration af converterens porte (se tabel B.1).

Bit7			ADCON0				Bit0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/ $\overline{DONE}$	———	ADON

Bit7			ADCON1				Bit0
———-	———-	ADFM	———-	PCFG3	PCFG2	PCFG1	PCFG0

Tabel B.1: ADCON0 og ADCON1 registrerne.

B.2 Konvertering

Dette afsnit indeholder en kronologisk gennemgang af de operationer, der udføres ved en konvertering.

B.2.1 Konfiguration af A/D converteren

- **Konfiguration af porte (ADCON1).** Det første, der skal gøres, er at få konfigureret diverse porte i ADCON1 registret, som skal benyttes til A/D converteren (Bit PCFG3-PCFG0).

- **Valg af A/D input porte (ADCON0).** Dernæst skiftes der til ADCON0 registret, for at få valgt det antal analoge inputs, der skal bruges til converteren (Bit CHS2-CHS0).
- **Valg af clock (ADCON0).** Der kan benyttes fire forskellige clockhastigheder til styring af konverteringen (Bit ADCS1-ADCS2):
  - $T_{AD} = 2T_{osc}$
  - $T_{AD} = 8T_{osc}$
  - $T_{AD} = 32T_{osc}$
  - T_{AD} = Intern RC oscillator

  T_{AD} er konverteringstiden pr. bit. RC er en intern clock, som stadig fungerer, når PICen er i SLEEP. For en PIC med clock frekvens over 1 [Mhz] er det nødvendig at være i SLEEP for at få en brugbar konvertering, når RC clock benyttes. RC clock har typisk en clock tid på 4 [μs] [Microchip-Midrange, 1997]. De 4 clock er specificeret i tabel B.2

Clock		Frekvens			
Operation	ADCS1/ADSC0	20 [MHz]	5 [MHz]	1.25 [MHz]	333.33 [kHz]
$2T_{OSC}$	00	**100 [ns]**	**400ns**	1.6 [μs]	6 [μs]
$8T_{OSC}$	01	**400 [ns]**	1.6 [μs]	6.4 [μs]	**24 [μs]**
$32T_{OSC}$	10	1.6 [μs]	6.4 [μs]	**25.6[μs]**	**96 [μs]**
RC	11	2-6 [μs]	2-6 [μs]	2-6 [μs]	2-6 [μs]

Tabel B.2: Clock varianter for A/D converteren.

De markerede tider er alle uden for den tilladte grænse for T_{AD}, enten for den minimal eller maksimal tilladte clock tid. Den maksimal tilladte tid skyldes, at kapacitorerne aflades under konverteringen, dermed er der fastsat en maksimal clock tid på 25[μs] for et pålidelig resultat under konverteringen. [Microchip-Midrange, 1997]

- **Initialisering af A/D modulet (ADCON0).** ADON er den bit, der starter converteren. Her vil en ny konvertering altid starte, medmindre der er brug for en anden konfiguration af converteren.

Konfiguration af A/D interrupt. Converteren sætter et interrupt flag, når konverteringen er afsluttet. De tre følgende bit bruges, når et interrupt skal konfigureres.

- **ADIF** sættes lav. A/D converterens interrupt flag bit, dette sættes høj, når konverteringen er færdig.
- **ADIE** sættes høj, for at tillade A/D interrupt.
- **GIE**, som er den globale interrupt bit, sættes høj.

Acquisition tiden. Acquisition tiden T_{ACQ} er den tid, der går, før converteren er klar til konvertering. Dvs., den tid det tager kapacitorerne at blive ladet fuldt op. Sættes konverteringen for hurtigt i gang vil det gå udover præcisionen, og resultatet vil blive unøjagtigt. T_{ACQ} kan udregnes på følgende måde:

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF} \tag{B.11}$$

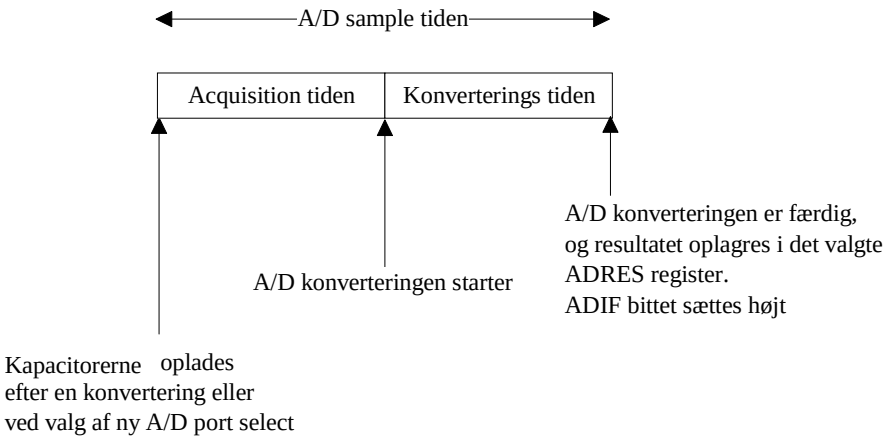
hvor T_{AMP} er tiden, der går, før comparatoren er stabil , T_C er tiden, det tager kapacitorerne at blive ladet op, og T_{COFF} er en temperatur koefficient, som først har en indvirkning ved 25[°].

Worst case (minimum) værdien for T_{ACQ} er 19.72 [μs] [Microchip-Midrange, 1997]. Beregnet med en indgangsimpedans på 10 [kΩ] og en temperatur på 50 [°].

Konverteringen startes(ADCON0). En 10 bit konvertering tager 11.5 T_{AD} før den er færdig [Microchip-16F87X, 2001]. Konverteringen startes ved at GO/ $\overline{DONE}$ bittet sættes høj (ADCON0), og når konverteringen er færdig, går bitten lav.

Samplingstiden. Den samlede samplingstid er acquisitionstiden adderet med konverteringstiden:

Dette er illustreret på figur B.10



Figur B.10: Den samlede sampletid.

Resultatet. Resultatet aflæses i de før omtalte registre ADRESH/ADRESL. Den absolutte fejlmargin for konverteringen er specificeret til $< \pm$ 1LSb, når $V_{REF} = V_{DD}$. Nøjagtigheden af konverteringen vil forværres, når V_{REF} divergerer fra V_{DD}. [Microchip-Midrange, 1997]

Næste konvertering. Før A/D converteren er klar til næste konvertering, går der 2 T_{AD}, hvor kapacitorerne ikke er tilgængelig på den valgte A/D input port [Microchip-16F87X, 2001].

Radiolink BIM-433-F

C

Formålet med dette afsnittet er at fremsætte en nærmere beskrivelse af BIM-433-F. I afsnittet vil det fremgå hvilke karakteristika radiomodulet besidder, i en mere hardwaremæssig henseende. Dette omfatter den fysiske benopsætning, rækkevidde og effektforbrug. Endvidere vil en beskrivelse af timingen i kommunikationen blive fremhævet med henblik på at opnå en korrekt dataoverførsel

C.1 Transceiver modul

Til den trådløse dataoverførsel benyttes to transceiver radiomoduler, BIM-433-F. En transceiver er i stand til at operere i half duplex mode, hvilket giver mulighed for både at sende og modtage data, i henholdsvis en transmit og receive mode. De to modes er ikke aktive samtidig, eftersom dette kræver et modul, som understøtter full duplex dataoverførsel. Radiomodulerne opererer ved udsendelse af UHF FM radiobølger med en frekvens på 433.920 [MHz], med en maksimal overførselshastighed af data på 40 rækkevidde på ca. 30 [m] uden antenne, og ca 120 [m] med en korrekt konstrueret antenne.

C.1.1 Konfiguration

BIM-433-F er opbygget således, at de er kompatible for direkte tilslutning til en microcontroller, idet de indeholder en FM-modulator samt FM-demodulator. Der anvendes 4 ben for tilslutning mellem modulerne og microcontrolleren. Til data som henholdsvis sendes og modtages, bruges de 2 databen TXD samt RXD. Til konfigurering af modes, anvendes de resterende 2 ben TX og RX, som er aktivt lave. Konfigurationen af de 2 modes, transmit og receive, ses af tabel C.1, hvor opsætningen af TX og RX benene fremgår:

Ben 15 TX	Ben 16 TX	Funktion
1	1	Power-down
1	0	Receive mode
0	1	Transmit mode
0	0	Test loop-back

Tabel C.1: Opsætning af BIM-433-F

Ud over de 4 omtalte ben, som er et minimum for at operere modulerne i half duplex mode, er der yderligere et ben med betegnelsen CD Carrier Detect. Dette ben er en aktiv lav udgang, som trækkes lav, når radiomodulet i receive mode opfanger en bærebølge, udsendt fra et tilsvarende modul. Den specifikke timing af CD signalet fremgår af det samlede timing diagram på figur C.1. Til trods for at modulerne er kompatible for direkte tilslutning til en microcontroller, vil det være hensigtsmæssigt at drive samtlige ind- og udgange gennem logiske CMOS enheder, for at sikre en tiltrækkelig impedans tilpasning. Vigtigt er især, at CD og RXD benet drives gennem en CMOS enhed, da disse har en høj udgangsimpedans på henholdsvis 50 [kΩ] og 10 [kΩ].[Radiometrix, 2001]

C.1.2 Data pakker

Når en mængde data ønskes transmitteret mellem to microcontrollere, skal de specifikke data sendes i datapakker gennem radiomodulerne. Pakkernes størrelse kan frit bestemmes og defineres af en enkelte bruger. Pakkerne sendes som asynkron seriel data både mellem radiomodulerne, og mellem radiomodul og microcontroller. Ved datatransmission sendes de serielle datapakker ofte på følgende form:

- **Preamble.** En preamble er en mængde data, som altid skal sendes forud for de reelle datapakker for at etablere og stabilisere forbindelsen.

- **Kontrol.** En eller flere bits angiver start- og slutplaceringen af de reelle datapakker i den afsendte datastreng. Yderligere kan informationer til dekodning omkring pakkelængde samt pakkeantal være angivet her.

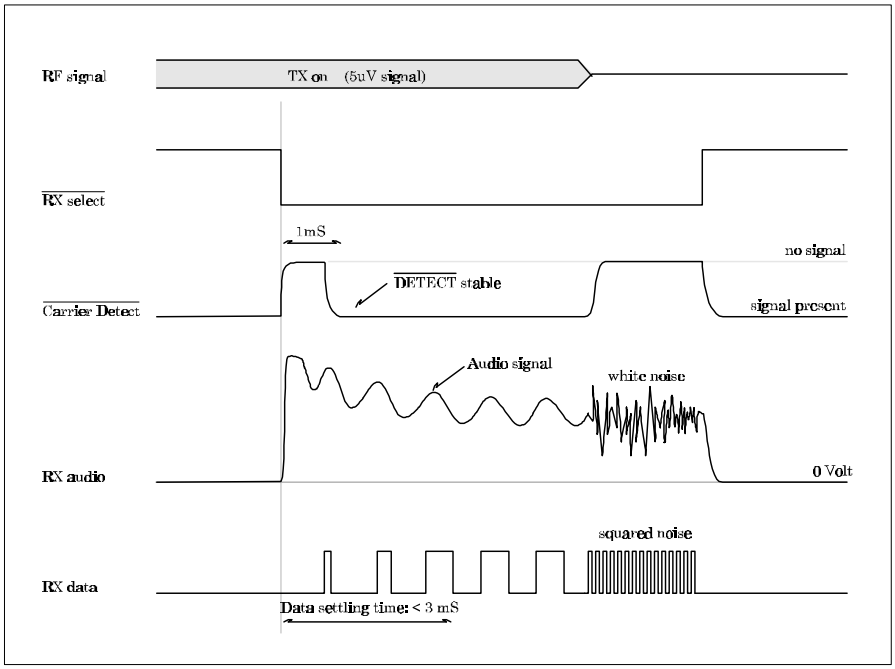
- **Adresse.** Hvis de sendte brugerdata skal placeres bestemte steder i hukommelse hos modtageren, kan adressen på den ønskede hukommelsescelle angives. Derudover kan modulnummeret indgå, hvis der anvendes flere moduler i netværket.

- **Data.** Den egentlige datamængde som lagres hos modtageren. Pakkens størrelse kan frit bestemmes, men bør maksimalt være 256 bits lang, for at mindske risikoen for fejl i pakken, og dermed et efterfølgende behov for en retransmission [Radiometrix, 2001].

C.1.3 Timing

For at sikre en pålidelig dataoverførsel har radiomodulerne en række krav til timing. Der skal tages hensyn til timingen mellem de forskellige ben og ved skift mellem de to modes. Når RX benet trækkes lavt, og modulet befinder sig i receive mode, vil CD benet først være aktiv lavt efter ca. 1 [ms] ved detektion af en bærebølge. De modtagede data på RXD benet vil ligeledes først være stabile efter 3 [ms]. Ved skift mellem transmit og receive mode og vice versa, vil radiomodulet befinde sig i stabil tilstand efter et tidsrum på ca. 1 [ms]. Et overblik over timingen af modulet i receive mode fremgår af timingdiagrammet på figur C.1,

hvor det også fremgår, hvorledes CD udgangen afhænger af timingen mellem kontrollinien RX og bærebølgen RF.



Figur C.1: Timingdiagram for radiomodul.

C.1.4 Krav til timingen

I forlængelse af de ovenstående timingparametre, er der yderligere nogle vigtige hensyn, som bør tages i betragtning for at opnå en fornuftig datatransmission mellem to moduler. Følgende tre krav skal som minimum være opfyldt, for at transceiveren overhovedet kan opfange og demodulere et signal.

- **Sendehastighed.** Under afsendelsen af data, skal tiden T mellem 2 transitioner opfylde følgende: 25 [μs] < T < 2 [ms]. Dette giver en sendehastighed på mellem 500 [bit/s] og 40 [kbit/s].

- **RX stabiliseringstid.** Når radiomodulet skifter til receive mode, vil der gå højst 4 [ms], før data på RXD benet vil være pålidelige. Derfor er det nødvendigt først at sende en mængde data, der ikke har nogen værdi, men blot skal sørge for at radiomodulet stabiliseres. En løsning på dette opnås ved at sende en såkaldt preamble bestående af sekvens af '10101010' i mindst 3 [ms]. Ved at øge denne periode til 5 [ms], opnås yderligere en større immunitet overfor interferens på RF benet.

- **Manchester kodning.** For at receiveren kan opfange og demodulere et UHF signal korrekt, er det nødvendigt, at der i signalet er en vis variation i bitmønsteret. En løsning vil være at kode det afsendte bitmønster så sekvensen indeholder et gennemsnitligt ens antal 1 og 0 bit over en periode på 4 [ms]. En løsning på dette kan være at anvende manchester kodning, som sikrer, at en datapakke sendes i et bitforhold på 50:50. Manchester kodning er en nødvendighed ved sendehastigheder omkring 40 [kbit/s] eller ved krav om en stor rækkevide. Under forhold, hvor rækkeviden og transmissionshastigheden ikke er afgørende, kan et 1:0 forhold på 30:70 tillades.

[Radiometrix, 2001]

C.1.5 Power-down mode

De anvendte radiomoduler er ideelle til applikationer, hvor et lavt strømforbrug er ønskeligt. Dette gælder især under forhold, hvor modulerne forsynes af et batteri. Under drift, hvor transceiveren befinder sig i enten transmit eller receive mode, er det gennemsnitlige strømforbrug ca. 12 [mA]. Efter gældende lovkrav må der i det frie frekvensbånd kun ske dataudveksling med en dutycycle på 10% [Telestyrelsen, 2000]. Dermed er det ikke nødvendigt, at et modul vedvarende står i receive mode og "lytter"efter et bærebølgesignal. BIM-433-F har den egenskab, at modulet kan sættes i power-down mode, hvormed strømforbruget reduceres til ca. 1 [mA]. [Radiometrix, 2001]

Power-down mode opnås ved sætte både TX og RX benene høje, som det fremgår af tabel *C*.1. For at detektere om en evt. bærebølge er inden for rækkevide, vil det være nødvendigt at "vække"radiomodulet i bestemte tidsintervaller. I forhold til kravsspecifikationen er der i dette projekt valgt at kontrollere for en evt. bærebølge for hver 90 [ms]. Når radiomodulet bliver vækket, samples der på CD benet i 5 [ms], idet der går 5 [ms] før en bærebølge kan detekteres.

I perioden på 5 [ms] skal CD benet samples. Hvis udgangen er lav, skal transceiveren forblive i receive mode. Hvis CD udgangen under samplingen ikke går lav, skal modulet returnere til power-down mode.

Med et wake up for hver 90 [ms], stilles der samtidig et krav til en preamble, som har en varighed på mindst 94 [ms], således at receiveren har mulighed for at vågne (<1 [ms]) og stabilisere RXD benet, efter RX benet er sat lavt (3 [ms]).

I^2C protokollen

D

Det følgende afsnit har til formål at beskrive I^2C protokollen. Indledningsvis beskrives de fysiske rammer omkring bussen, hvorefter en nærmere fremhævelse af selve protokolstandarden vil blive gennemgået. Herunder timingen samt de definerede bussekvenser.

D.1 I^2C bussen

I^2C interfacet er en standard inden for seriel buskommunikation udviklet af Philips, der tillader bidirektional udvekling af data over 2 linier der tilsammen udgør bussen. De 2 linier betegnes SCL (Seriel Clock Line) og SDA (Seriel Data Line), som forbindes mellem enhedernes I/O porte. I^2C protokollen giver mulighed for dataudveksling efter master/slave princippet, der har en maksimal kommunikationshastighed fra 100 [kbit/s] til 3,4 [Mbit/s] high speed mode, efter en optimering i 1998. Fast mode er dog den mest anvendte standard med en øvre hastighed på 400 [kbit/s], udviklet i 1993. [Katzen, 2000]

Master/slave princippet operere ved at der fra masterenheden sendes en adresse på bussen, som tilsvarer en slaveenhed i systemet, hvorefter der oprettes forbindelse mellem de to enheder. En masterenhed kontrollerer bussen ved at generere en clock cycle samt de relaterede start- og stopbetingelser. Dermed kan masteren kommunikere med et given antal slaveenheder, enkeltvis eller samtlige på en gang. Masterenheden kontrollerer endvidere kommunikationsretningen, ved at konfigurere systemet til enten mastertransmit eller masterreceive mode.

Under kommunikation med I^2C protokollen er der ingen begrænsninger i mængden af data som kan transmitteres mellem de kompatible enheder. Den serielle data sendes en byte ad gangen, og når adressen på slaveenhden er sendt, kan masteren bestemme den ønskede datamængde.

I^2C protokollen understøtter endvidere mulighed for multimastermode, hvor to eller flere masterenheder kan inddrages i systemet. Det ideelle ved I^2C protokollen er at enheder frit kan tilføjes eller fjernes fra bussen, uden at påvirke det øvrige system.

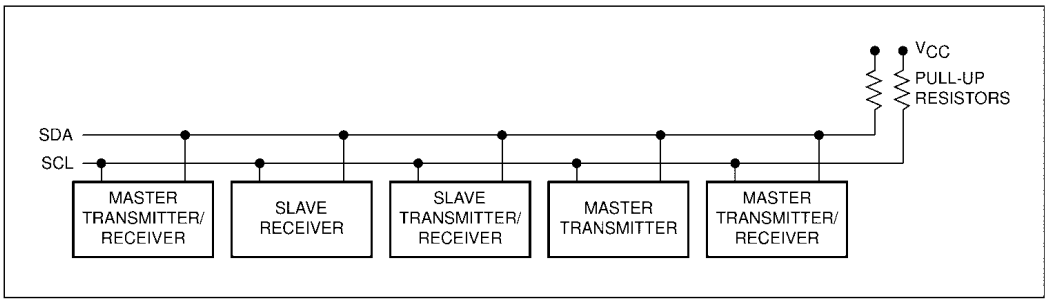
D.2 I²C bus specifikation

For at opnå en nærmere grundlæggende forståelse for I²C protokollens opbygning, gennemgås princippet bag kommunikationen med I²C.

Når bussen befinder sig i idle state, der er defineret, når der ikke sendes data, og bussen er ledig, skal bussens datalinie (SDA) og clocklinie (SCL) være logisk høje. Da bussen i idle state skal være høj, forsynes de to linier SCL og SDA med en pull up modstand hver på 10[kΩ] til V_{CC} (se figur D.1). I/O portene, som tilsluttes SDA og SCL, skal være open drain output, og forbindes med andre open drain outputs på bussen.

Hvis outputtet fra en enkelt enhed trækkes logisk lav, vil bussen trækkes til jord og de øvrige enheder vil betragte signalet på bussen som logisk lav.

For at lette forståelsen i de nedenstående beskrivelser, henvises til figur D.2 og D.3.



Figur D.1: I²C bus med pull up modstande.

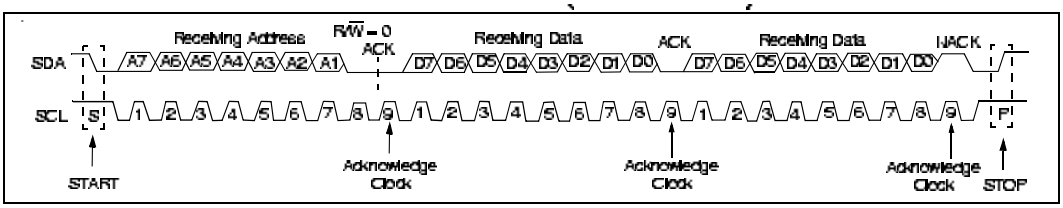
Start- og stopsekvenser. I henhold til I²C specifikationen skal alle dataændringer på SDA linien foretages, mens clocken på SCL er lav. Det er endvidere vigtigt, at data på SDA linien er stabile, mens clocken er høj. Denne restriktion tillader, at to vigtige start og stop sekvenser kan detekteres. En startsekvens er defineret ved en høj til lav transition på SDA linien, mens SCL holdes høj, og indikerer til alle slaveenheder i systemet, at en adressebyte er ved at blive sendt. En stopsekvens er defineret ved en lav til høj transition på SDA linien, mens SCL holdes høj. Dette indikerer, at transmissionen er færdig, og SDA linien termineres, sådan at slaveenheden kan resette hardwaren og vente på en ny startsekvens.

Adressering. Master/slave princippet fungerer ved adressering af de respektive slaveenheder. Forud for hver transmittering af data, sendes en slaveadresse ud på bussen. I²C protokollen understøtter enten 7 eller 10-bits adressering og betyder, at der i princippet er mulighed for servicering af 128 eller 1024 slaveenheder. Dog er visse adressekombinationer reserveret til specielle formål, som f.eks. general call adressering. Eftersom data sendes 8-bit ad gangen, vil det være nødvendigt at sende 10-bit adressering over 2 byte. I 7-bits mode

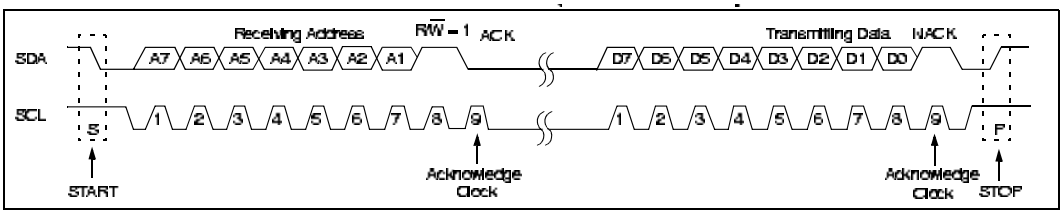
indeholder det sidste bit information om, hvorvidt der ønskes at skrive eller læse fra slaveenheden ved ændring af $R/\overline{W}$ bitten. Når en slaveenhed genkender sin egen adresse, vil den generere et acknowledge signal til masterenheden.

Acknowledge. I²C protokollen opererer med handshake mekanismen. Efter transmittering af den 8. bit, generes der en 9. clockpuls af masteren. Under denne periode frigøres SDA linien, og receiveenheden kan returnere et acknowledge signal (ACK) til transmitenheden. Et acknowledge indikeres ved, at SDA sættes lav og genereres, når en byte er sendt og modtaget korrekt. Omvendt forbliver SDA linien høj, hvis data ikke er modtaget korrekt, hvilket betegnes not acknowledge (NACK). I sådanne tilfælde må brugeren vælge om transmitteringen evt. skal stoppe, genstarte eller lignende. Med returnering af et ACK henholdsvis NACK, kan det undgås at en større mængde data går tabt under fejltransmittering, og efterfølgende skal sendes igen. Et NACK benyttes også af masteren i receive mode til at angive slutningen på en transmittering, når den sidste byte er modtaget.

Repeated start. En sidste funktion som benyttes i I²C protokollen er repeated start condition (RS). Repeated start er den samme sekvens som en startsekvens, men kan genereres under en transmission uden en forudgående stopsekvens. Ofte bruges RS sekvenser ved skift mellem write og read mode, hvor slaveadressen sendes efterfølgende med ændret $R/\overline{W}$ state. Ved at generere et repeated start sekvens sikres, at masteren ikke mister kontrollen over bussen, hvilket ville risikeres, hvis en stop/start sekvens blev brugt. Dette er især gældende under multimastermode.



Figur D.2: Typisk I²C write transmission.



Figur D.3: Typisk I²C read transmission.

Data arbitration. Ved anvendelse af multimastermode vil der være stor risiko for buskollision, hvis to eller flere masterenheder forsøger at opnå kontrol over bussen samtidigt. Derfor bruger både clock- og databenet den såkaldte wired-and funktion til at kontrollere bussen. Ved at overvåge clock- og databenet forventer masteren, at den logiske tilstand på benet tilsvarer, hvad der oprindeligt er sendt. Hvis en anden enhed forsøger at kontrollere bussen samtidig, vil dette opdages, hvis f.eks. et ben er logisk højt men forventes logisk lavt. En sådan hændelse kaldes også for lost arbitration, og ved indtræffelse skal masteren opgive kontrollen med bussen. Wired-and funktion bruges altså til at kontrollere og overvåge clock på SCL benet samt at opnå data arbitration på SDA-benet.

DS1621 - Temperaturmåling

E

Til PIC16F877 er der tilsluttet et digital termoter. Den valgte sensor er Dallas Semiconductors model DS1621, der understøtter I^2C bus.

E.1 Karakteristik af DS1621

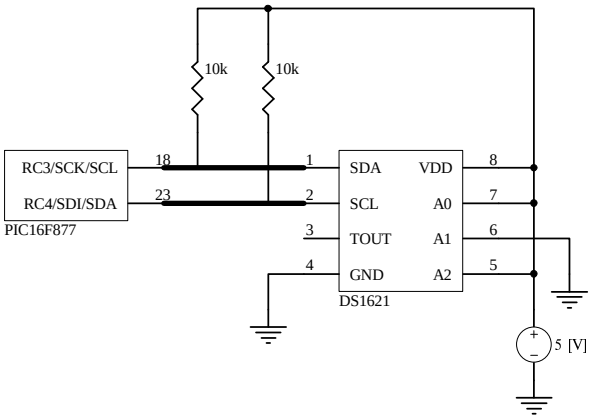
Dallas DS1621 er en IC, der ikke behøver yderligere tilslutninger end forsyningsspænding for at fungere. I chippens firmware er der indbygget A/D converter, EEPROM hukommelse og termostat for indstilling af advarsel ved temperatur over eller under brugerindstillet grænseværdier. Følgende karakteristika er gældende for DS1621:[Dallas-Semiconductors, 1999]

- Temperatur målingsområde: $-55[°C] \longrightarrow 125[°C]$.
- Opløsning på $0.5[°C]$.
- 2 wire serial data bus, efter I^2C standard.
- Understøtter standard mode (100[kHz]) og fast mode (400[kHz]).
- Termostat alarmering.
- Forsyningsspænding $2.7[V] \longrightarrow 5.5[V]$
- Lavt strømforbrug ved standby $1[\mu A]$
- Temperatur konverteringstid, under $1[s]$

E.2 Elektrisk opsætning

Angivelse af slave adresse foretages via $A0, A1, A2$ ved deres logiske tilstand. DS1621 har en fast prefix adresse på 1001, de fire A pins sættes til 101, hvormed slave adressen bliver

1001101 $(04Dh)$. To pull up modstande på 10[kΩ] forbindes mellem V_{DD} og I^{2}C bussens linier.[Dallas-Semiconductors, 1999]



Figur E.1: Tilslutning af DS1621 til PIC16F877.

Figur E.1 viser den elektriske tilslutning af temperatur chippen DS1621 og PIC16F877 til I^{2}C bussen.

E.3 Timing

I projektet anvendes I^{2}C bussen til opkobling af DS1621. Data, der sendes over denne bus, skal overholde timingen, der er defineret i protokollen for I^{2}C. Protokollen overholdes af DS1621, idet den er I^{2}C kompatibel, hvorfor timing diagrammer ikke medtages.

E.4 Kommunikation mellem DS1621 og PIC16F877

I^{2}C kræver, at minimum én enhed på bussen er master. PIC16F877 vælges til master i dette system og DS1621 til slave. Masteren administrerer kommunikationen på bussen, hvilket klares i PICens MSSP modul.

Konfigurering. DS1621 er kun istand til at operere som slave. Opsætning af temperaturmålerens forskellige funktioner skal foretages inden, temperaturudlæsningen kan begynde. Konfigurationsværdien latches ind i et internt register via kommandoen 'Access Config' (0ACh).

På forhånd vælges, at termostat alarmen ikke benyttes. Bitpolaritet hvor 1 = high, samt uafbrudt temperatur konvertering. Den indlæste konfigurationsstrengen bliver dermed 10001010b = 08Ah.

Opsætningen skal kun foretages én gang, hvorfor der laves en initialiseringsrutine, der sender opsætningsstrengen samt starter konverteringen. Rutinen kaldes under opstart af PICen, og ændres ikke efterfølgende.

I det pågældende projekt vælges endvidere en temperaturopløsning på $1[°C]$, hvormed temperaturaflæsningen blot behøver MSB af de to temperaturbytes der er til rådighed.

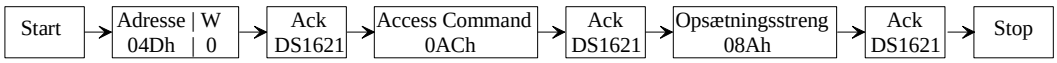
E.5 Gennemgang af kommunikationsprocedure for DS1621

DS1621 opererer i to modes bestemmende for, om den skal modtage eller afsende data. Ved modtagelse sættes DS1621 i receive mode Transmit mode opnås, når enheden latcher data ud på bussen. Valget af mode foretages under adressering af den valgte enhed, hvor sidste bit af adressen ($R/\overline{W}$) sættes højt eller lavt.

For hver byte der latches ud på bussen, skal receiveren kvittere med et acknowledge bit (ACK), hvis overførslen er korrekt og forstået. Ved fejl sættes not acknowledge (NACK). Dette gælder dog ikke, når master skal afslutte en modtagelse, hvor der sendes et NACK.

Ved alle transmissioner skal start og stop sekvenserne sættes, således at det vides, hvornår bussen er ledig for andre enheder der er tilkoblet systemet.

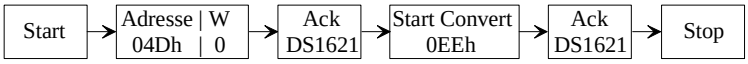
Initialiseringen af DS1621 fremgår af nedenstående figur:



Figur E.2: Flow af initialiseringsproceduren.

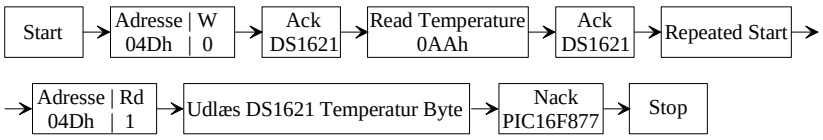
Af figur E.2 fremgår det, hvorledes opsætningen af DS1621 foregår. Efter valg af opsætningskommandoen sendes den byte, der indeholder de forskellige parametre, DS1621 benytter.

Aktiveringen af temperatur målingen kræver, at der sendes en 'start convert' kommando, hvilket fremgår af figur E.3



Figur E.3: Procedure for konverteringsstart.

Ved udlæsning af temperaturdata ændres DS1621 til transmit mode, og masteren, dvs. PI-Cen, skifter til receive mode.



Figur E.4: Temperatur udlæsningsprocedure.

Af figur E.4 fremgår, at der kun udlæses én byte fra DS1621, grundet valget af $1[°C]$ opløsning. Ønskes større temperatur opløsning, skal LSB af temperaturdata også overføres. Derved sendes to fortløbende bytes. Mellem hver byte skal masteren sende et acknowledge.

MatLab kildekode

F

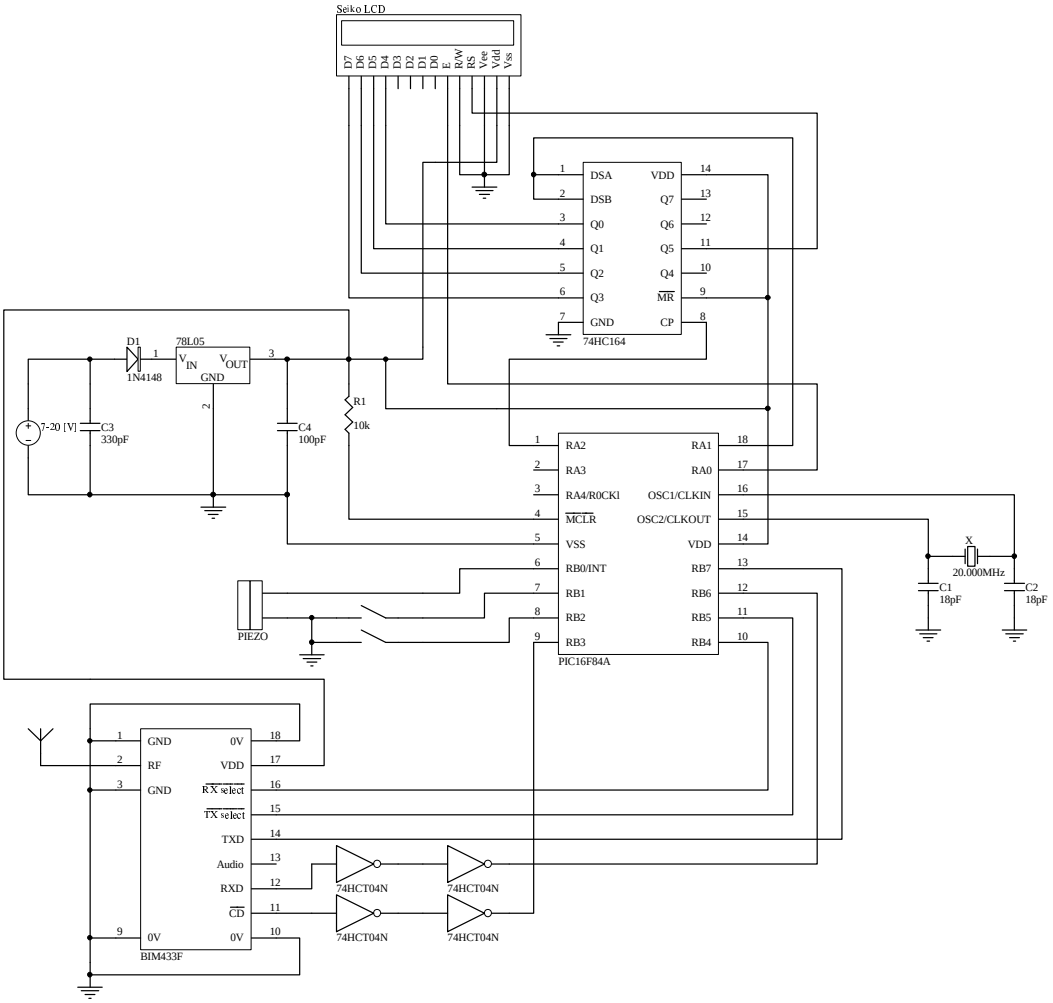
F.1 Bestemmelse af generator og check matricer

```
%Generér alle 5-bit datavektorer med hamming vægt > 2
A=0;
x=5;
for i = 1:(2^x-1);
   w=0;
   P = de2bi(i,x);
   for j = 1:x;
      w=w+P(1,j);
   end;
   if (w>2);
      A=[A;i];
   end;
end;
A(1,:)=[];

%Generér mulige 16-bit datavektorer med hamming vægt = 11
B=0;
x=16;
for i = 1:(2^x-1);
   w=0;
   P = de2bi(i,x);
   for j = 1:x;
      w=w+P(1,j);
   end;
   if (w==11);
      B=[B;i];
   end;
end;
B(1,:)=[];

H = eye(11);

for i = 1:length(B);
    S = de2bi(B(i,1),16);   % Udvælg det i. 16-bit/11-vægts ord
    G = A.* transpose(S);   % Gange overflødige elementer ud.
    K = 0;
```

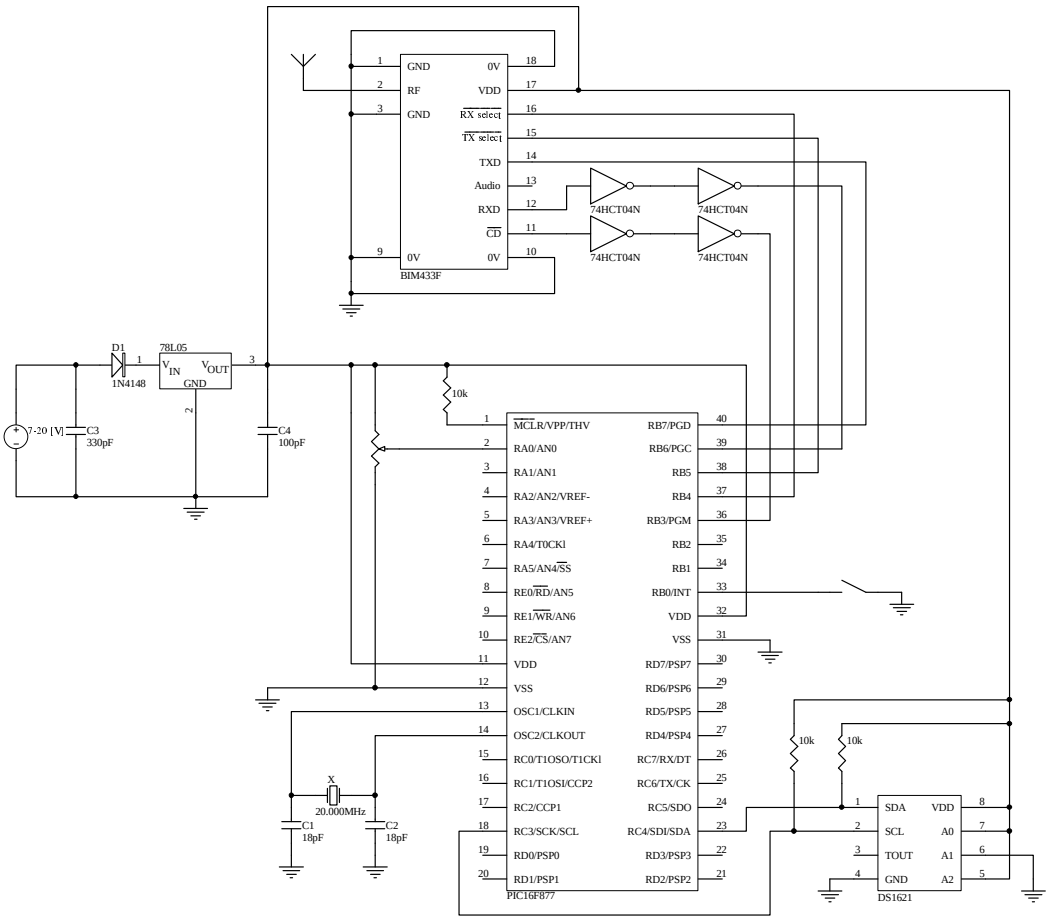
```
37      for j = 1:16;           % Fjern nul elementer
38          if (G(j,1)~=0);
39              K = [K;G(j,1)];
40          end;
41      end;
42      K(1,:)=[];              % Paritetsmatrix er bestemt
43      G=de2bi(K);
44      G = [H G];              % Definer generator udfra paritet og I11x11
45      dmin = 100;
46      for j = 1:(2^11-1);     % Udregn kodens dmin
47          C = de2bi(j,11);    % C er ukodet 11-bit datavektor
48          D = mod((C*G),2);   % D er modulo-2 produktet CG
49          w = 0;
50          for k = 1:16;       % Udregn D's vægt
51              w = w + D(1,k);
52          end;
53          if (w < dmin);      % If løkke finder mindste vægt for G
54              dmin = w;
55          end;
56      end;
57      if (dmin == 4);         % Hvis dmin = 4 -> Korrekt generator gemmes
58          save generator G;
59      end;
60  end;
61  G = transpose(G)
62  H = null(transpose(G),'r'); % Udregn H som nulrummet af G transponeret
63  HT = transpose(mod(H,2))
```

Elektriske diagrammer

G



Figur G.1: Elektrisk diagram over master systemet (fjernbetjeningen)



Figur G.2: Elektrisk diagram over slave systemet (bilen)

Litteratur

Dallas-Semiconductors [1999]. *Digital Thermometer and Thermostat DS1621*, Dallas Semiconductor.

Ebert, H. [2000]. Informationsteori.
URL: *HTTP://www.kom.auc.dk/ heb/kurser/infnote.pdf*

Julie [1994]. History of the remote control.
URL: *HTTP://www.modellbahnott.com/tqpage/ihistory.html*

Katzen, S. [2000]. *The Quintessential PIC Microcontroller*, Springer-Verlag.

Kugelstadt, T. [2000]. The operation of the sar-adc based on charge redistribution.
URL: *HTTP://www-s.ti.com/sc/psheets/slyt012b/slyt012b.pdf*

Lay, D. C. [1996]. *Linear Algebra and its Applications*, 2 edn, Addison-Wesley, USA. ISBN:0-20-134774-1.

Microchip-16F84A [1998]. *PIC16F84A 18-pin Enhanced Flash/EEPROM 8-Bit MCU Data Sheet*, Microchip Technology Inc., USA.

Microchip-16F87X [2001]. *PIC16F87X Data Sheet*, Microchip Technology Inc., USA.

Microchip-Midrange [1997]. *PICmicro^{tm} Mid-Range MCU Family Reference Manual*, Microchip Technology Inc., USA.

National-Semiconductors [2000]. *LM78LXX Series*, National, USA.

Pretzel, O. [1992]. *Error-Correcting Codes and Finite Fields*, student edn, Oxford University Press, Oxford, GB. ISBN:0-19-269067-1.

Radiometrix [2001]. *Low Power UHF Data Transceiver Module*, Radiometrix Ltd, England.

Seiko-Instruments [1998]. *Character LCD Modules*, Seiko.

Telestyrelsen [2000]. *Dansk radio grænseflade 00 032*, Telestyrelsen.