1 Inledning

Denna rapport är en dokumentation av projekt ”J.O.R.D” vilket har genomförts som examination i kursen Mikrodatorprojekt, TSIU51. Projektet innebär att en klockradiosändare, med uppgift att skicka ut datum och tid till klockradioapparater, skall konstrueras. Sändningen skall skickas ut enligt DCF77-standarden, vilket är en av de standarder som idag används för att skicka ut information i form av radiovågor till radioklockor. DCF77-standarden finns beskriven i kapitel 5.

2 Vad är en radioklocka?

En radioklocka är ett ur som får datum och tidsangivelse från radiovågor. Dessa radiovågor innehåller den nödvändiga tidsinformationen i form av ett bitmönster som ställer klockans interna digitala urverk. Detta är väldigt användbart då användaren inte själv behöver ställa tiden, samt att detta sker med en extrem precision.

3 Beskrivning och kravspecifikation av projektet

Projektet är att konstruera en apparat som skall kunna skicka ut datum och tidsangivelser i form av radiovågor.
De grundkrav som anger ramen för hela projektet är:

- Apparaten skall kunna sätta tid och datum helt godtyckligt.
- Klockans noggrannhet skall vara bättre än 60 sekunder per dygn.
- Apparatens räckvidd skall vara mindre än 20 cm.

Utöver detta har gruppen valt att göra ett användarvänligt menysystem där det tydligt framgår vilken inställning användaren för tillfället behandlar.

4 Bakgrund

Inför projektets start gick ett flertal föreläsningar med genomgångar av hur baskomponenterna, såsom, processor, display, tryckknapar, vridreglage med mera, fungerar och är konstruerade. Mycket fokus lades på processorn och den LCD-display vilken skulle användas. Föreläsningarna omfattade också mycket exempelkod för att visa hur olika grundmoment kan utföras, vilket under konstruktionens gång visat sig vara väldigt användbart. Det gavs också 2 introducerande laborationer där möjligheten att bekanta sig med både utvecklingsmiljönAtmel Studio och hårdvaran fanns att tillgå.

5 Beskrivning av DCF77-standarden

Klockradiosändaren som utgör detta projekt är konstruerat utifrån DCF77-standarden vilket är den standard som används för sändning av information i radiovågor till bland annat radioklockor och annan apparatur som får sin tidsinformation och datumangivelse via denna sändningsfrekvens. DCF77 står för, (D = Deutschland, C = långvågssignal, F = Frankfurt, 77 = frekvens 77,5 kHz).

För att beräkna den tid som skall skickas från sändare i Tyskland används ett atomur.
Atomuret fungerar genom att mätningar görs på svängningstiden hos en cesiumatom. Informationen behandlas innan det skickas vidare för sändning. Nedanstående diagram illustrerar hur en sändning enligt denna standard går till. I diagrammet visas en sändning vilken motsvarar en minut av DCF77 signalen.

Sekund	Bit	Kommentar
0		
1 – 14		De första 14 bitarna är för väderinformation
15		1 om reservantenn används vid sändning
16		1 om förvarning om sommartid
17		Bitar som anger normaltid eller sommartid 01 vid normaltid, 10 vid sommartid
18		
19		1 om förvarning om skottsekund skall sändas
20	Startbit	Startbit för sändning, denna bit är alltid 1
21 – 27		Tid: Minuter, (bitvärde, 1, 2, 4, 8, 10, 20, 40)
28	Paritetsbit #1	Jämn paritet
29 – 34		Tid: Timmar, (bitvärde, 1, 2, 4, 8, 10, 20)
35	Paritetsbit #2	Jämn paritet
36 – 41		Tid: Dag, (bitvärde, 1, 2, 4, 8, 10, 20)
42 – 44		Tid: Veckodag, (bitvärde, 1, 2, 4)
45 – 49		Tid: Månad, (bitvärde, 1, 2, 4, 8, 10)
50 – 57		Tid: År, (bitvärde, 1, 2, 4, 8, 10, 20, 40, 80)
58	Paritetsbit #3	Jämn paritet
59		

Tabell:1 Diagram över de bitmönster som en minut av sändning innehåller

Bitmönstret för en minuts sändning visas tydligt i tabell 1, men något som är värt att nämna är de 3 paritetsbitar som finns i sändningen. Paritetsbitarna är kontrollbitar som klockans mottagare använder sig av när den verifierar trovärdigheten av den inkommande signalen. Paritetsbitarnas uppgift är att väga upp till ett jämnt antal ettor om så skulle behövas från den föregående paritetsbiten. En paritetsbit sätts till ett endas om det varit ett ojämnt antal ettor i föregående delsändning,räknat ifrån föregående paritetsbit alternativt från starten av en sändning.

6.1 Specifikation av hårdvara

För att genomföra detta projekt erfordrades det en mängd olika hårdvara.
Nedan följer en lista överden hårdvara som har används.

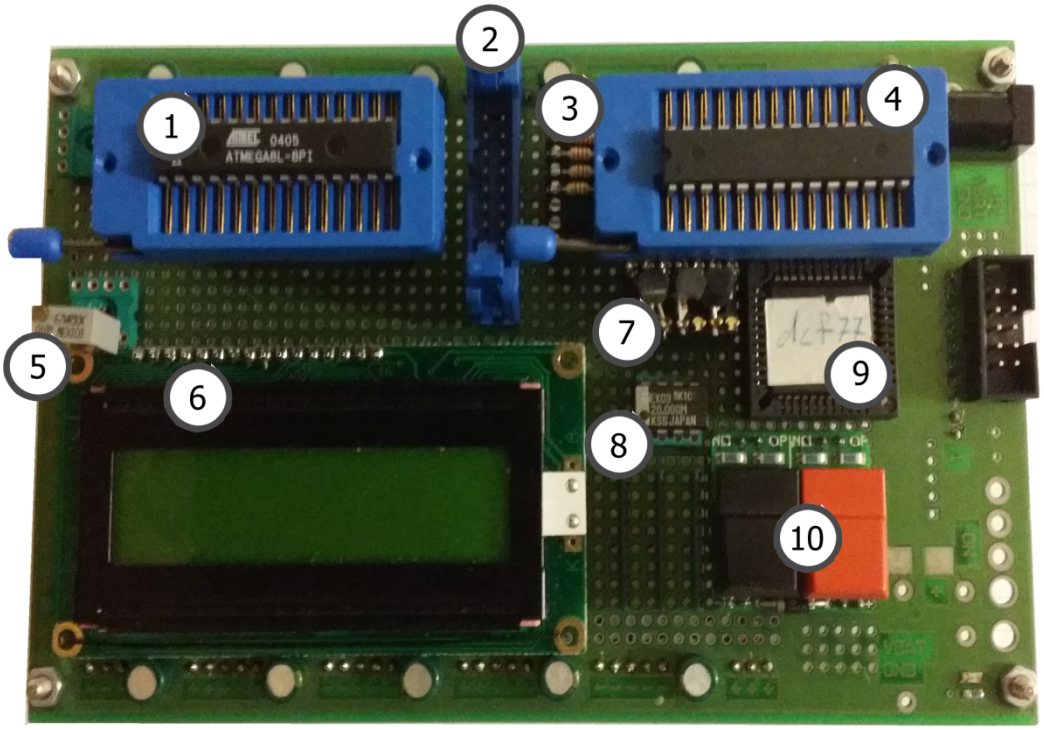
- 1 st LCD 16x2 JM162A
- 2 st mikroprocessorer av typAtmega8
- 1 st potentiometer (Används för att ställa displayens konstrast)
- 1 st CPLD (Frekvensgenererare)
- 1 st modulator (Används för amplitudmodulering av antennsignal)
- 2 st tryckknappar (Används som reset-knapp samt startknapp av sändningen)
- 2 st skruvreglage (Används vid menyn)
- 4 st motstånd 10 kΩ (Används i kopplingen mellan vridreglagen och processor 1)
- 1 st EXO3 (Kristalloscillator)

Övrig utrustning som har använts under konstruktionen av kretsen.

- Virningspistol
- Virningstråd
- Pincett
- Logikanalysator
- Trådavskalare

Här nedan visas en översiktsbild av kretsen och dess färdigmonterade hårdvara.

1 Mikroprocessor #1
2 Kontakt
3 Motstånd
4 Mikroprocessor #2
5 Potentiometer
6 Display 16x2
7 Modulator
8 Exo3 Kristalloscillator
9 PCLD DCF77
10 Tryckknappar

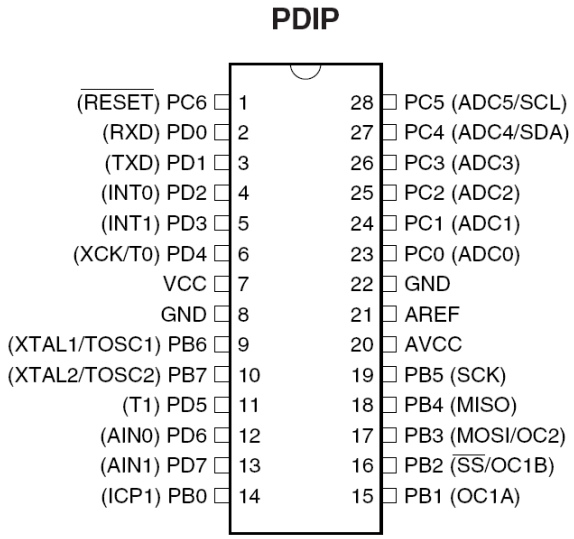


Figur: 1 Överblick av uppkopplingen

6.2 Processor, Atmega8

I detta projekt har 2 st processorer använts. Processorerna är av modellen Atmega8.

Mikroprocessorerna även kallade mikrokontrollers som användes i projektet var av typ Atmega8 en processor av RISC-typ utvecklad av det amerikanska företaget Atmel. En mikrocontroller eller mikroprocessor utgör själva kärnan i ett mikrodatorprojekt. Under den långa resan mot en slutlig produkt kommer processorerna vara spindlarna i nätet som knyter samma all övrig hårdvara och via externa signaler styr informationsflödet i de digitala kretsar av sladdar som utgör projektplattan. Allt detta åstadkoms genom programmerings i programmeringsspråket assembler för Atmega8 processorn.



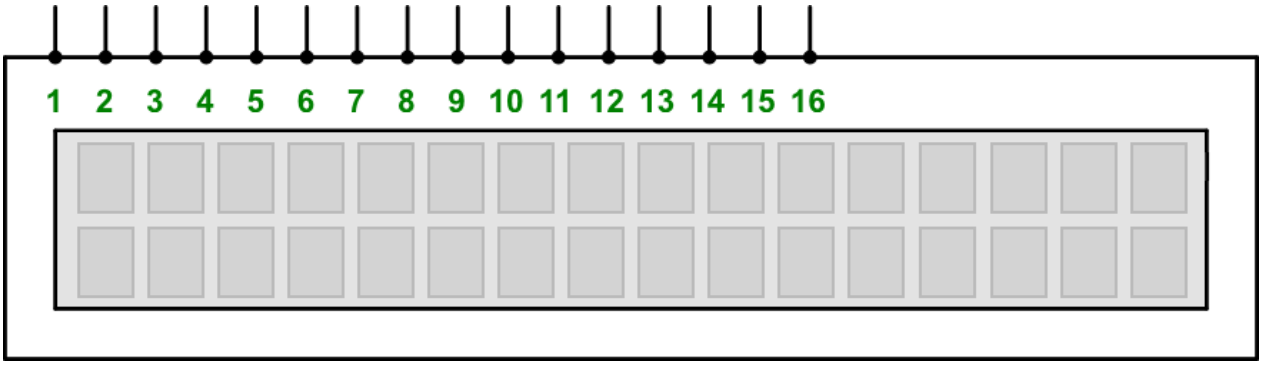
Figur: 2 Mikroprocessor Atmega8

För att Atmega8 skall fungera enligt specifikationerna i databladet kräves en spänning på 2,7 V till 5,5 V och drar ca 3,6 mA när processorn är igång och arbetar. Atmega 8 har vidare förmågan att gå i maximalt 16 Mhz via intern klocka men har också möjlighet att gå via en extern klocka, en möjlighet vilken utnyttjas i projektet med EX03 kristallen. Processorn är också utrustat med totalt 28 utpinnar, vilka pinnar som utnyttjas och hur dessa användes finns vidare beskrivet i kopplingsschemat.

6.3 EXO3 Kristalloscillator

Den kristalloscillator som används som frekvensgenererare heter EXO3 och genererar grundfrekvenser till övriga komponenter i konstruktionen. I detta fall genererar den 2 frekvenser, en på 20 MHz och den andra på 10 MHz. Hur dessa frekvenser används beskrivs närmare i kapitel 7.

6.4 Display LCD 16x2



Figur: 3 LCD Display 16x2

LCD-skärmen som användes var JM162A vilket är en karaktärsskrivande punktmatris LCD modul. LCD-skärmen tar in 8 bitar data för att tolka vilket tecken som skall visas på LCD-skärmen enligt en tabell som finns i LCD-skärmens minne. Detta minne är även programmerbart, för de gånger ett tecken som inte finns i minnet behöver skrivas ut. När LCD-skärmen får strömförsörjning måste den först initieras, för att den skall fungera. Detta utförs i flera steg, där många av databitarna används för att ställa in varje steg. Efter att bitarna har blivit mottagna behöver LCD-skärmen bearbeta denna information, vilket tar olika lång tid beroende på vilket steg i initieringsprocessen LCD-skärmen befinner sig på. Om korrekt initiering inte inväntas, resulterar det i att LCD-skärmen inte fungerar alls. Vid skrivning till LCD-skärmen kommer den behöva tecknet som skall skivas ut hämtas från minnet, under tiden kan flera tecken inte skrivas.

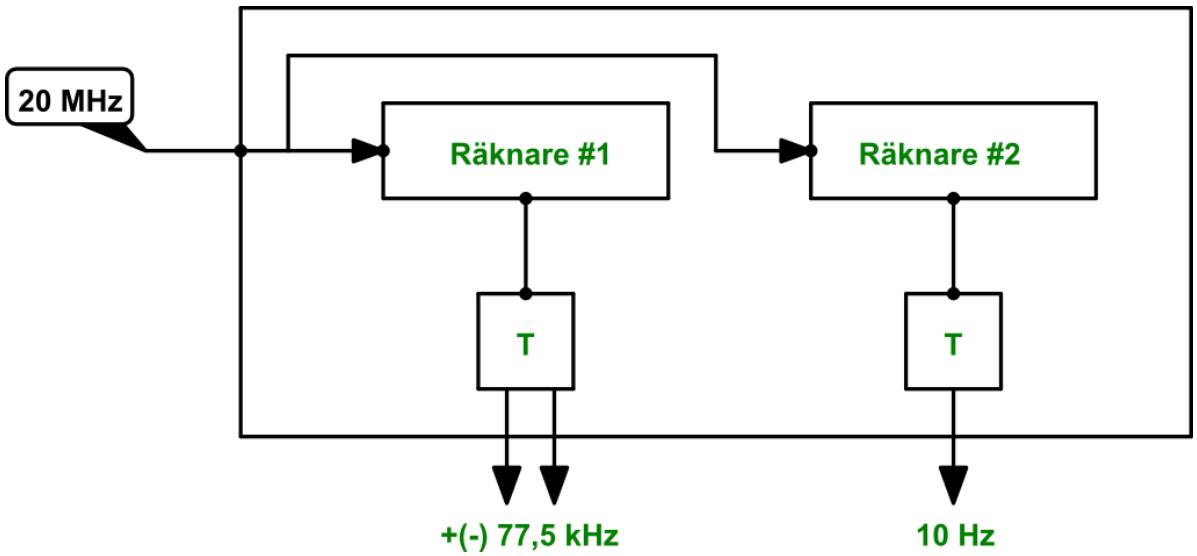
Utseendet av LCD-skärmen visas i figur 3, där dess ingångar och utgångar samt teckenplaceringen visas tydligt.

Pin	Funktion
1	GND
2	Vdd
3	Kontrastpotentiometer
4	Registerval, (H=Data, L=Instruktion)
5	Läsa/Skriva
6	Aktivera
7	Data bit 0
8	Data bit 1
9	Data bit 2
10	Data bit 3
11	Data bit 4
12	Data bit 5
13	Data bit 6
14	Data bit 7
15	Bakgrundsljus, Anod
16	Bakgrundsljus, Katod

Tabell: 2 Förklaring till figur 3

6.5 CPLD DCF77

Här under visas en förenklad grafisk tolkning av hur CPLD-komponenten fungerar i praktiken.



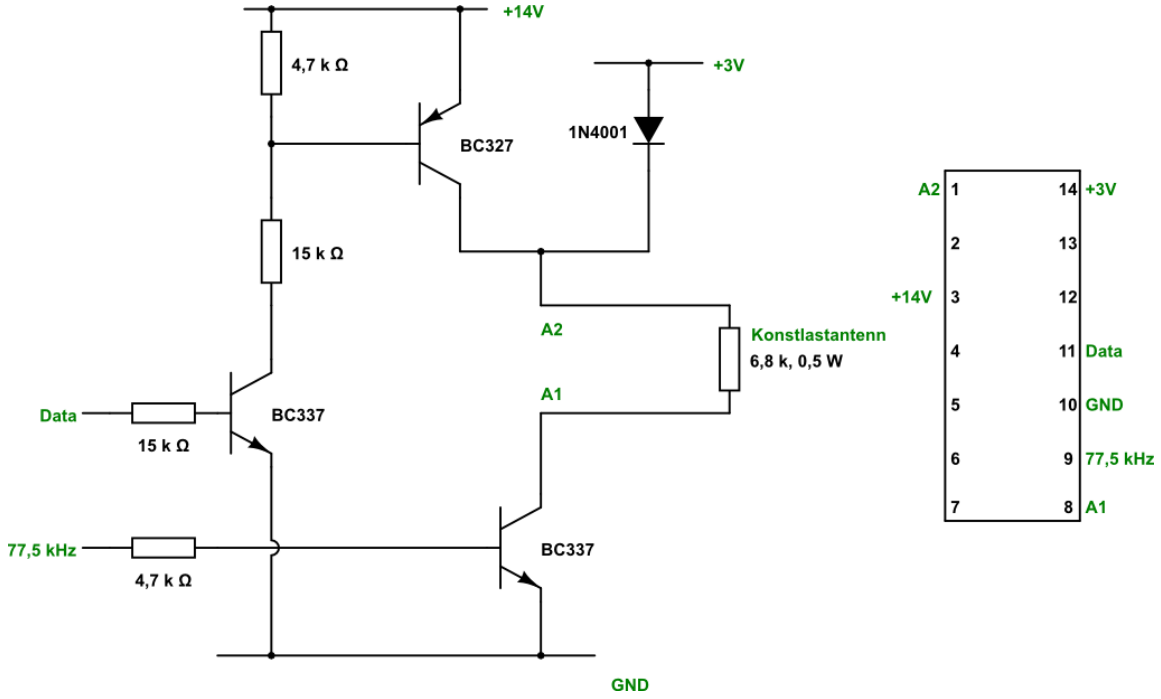
Figur: 4 Grafisk beskrivning av hur CPLD-komponenten är uppbyggd

Som figur 4 tydligt visar får CPLD-komponenten en ingående signal på 20 MHz som sedan skickas vidare till två separata räknare. Räknarnas uppgift är att skala den ursprungliga signalen för att modelera den till önskad utfrekvens. Räknarna består av kombinatoriska nät samt ett mängd d-vippor som är programmerbara, för att få önskvärda utsignaler. För att de båda signaler som skickas ut från CPLD-komponenten skall vara balanserade fyrkantsvågor skickas de från respektive räknare vidare genom var sin t-vippa, innan signalen skickas vidare ifrån CPLD-komponenten. T-vippan halverar frekvensen som kommer ut från räknarna innan signalen skicka ut ur CPLD-komponenten.

Hur dessa utgående signaler används i konstruktionen behandlas i kapitel 7.

6.6 Modulator

Mycket av signalbehandlingen sker hos modulatorn, vilket kommet att tas upp i detta kapitel.



Figur: 5 Schema över hur signalmodulatorn är konstruerad

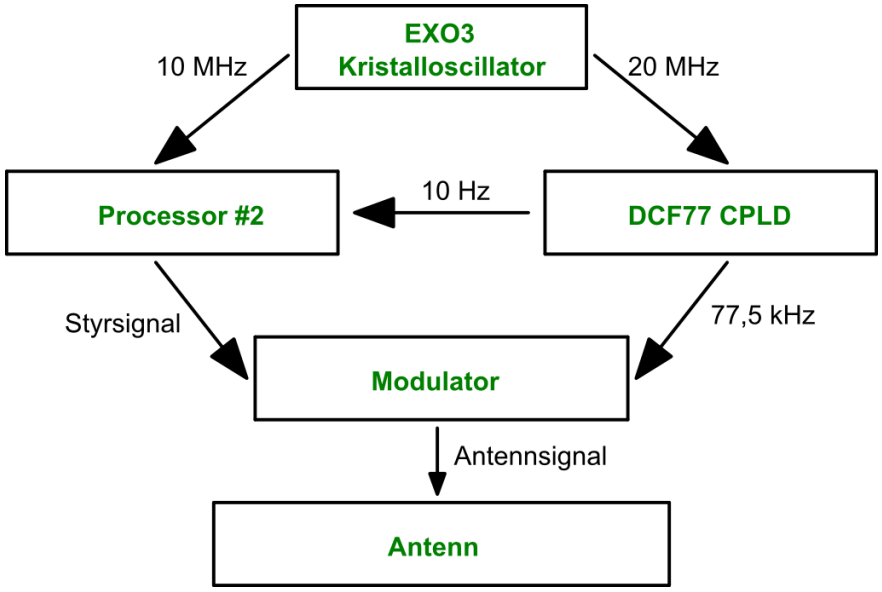
Modulatorn fungerar genom att en ingående signal med frekvensen 77,5 kHz skickas från CPLD-komponenten och in i modulatorn på pin 9. Samtidigt som detta sker skickas data i form av den klocktid och datum vilket ämnasbli sänt in på port 11 från processorn. Dessa data består antingen av en hög eller av en låg signal, alltså en etta respektive en nolla. Beroende på om den ingående datasignalen är hög eller låg kommer en spänning på 3 volt eller en spänning på 14 volt att skickas till antennen. Det är detta som gör att den utgående signalen blir av rätt modulering och signalstyrka.

Den antenn som används har en räckvidd på cirka 7 cm och består av ett motstånd på 6,8 k, 0,5 W. Antennen kopplas in direkt på modulatorns utgående portar för antennsignalen, A1 och A2, se figur 5.

Även då projektet inte var helt färdigt för att skicka ut en korrekt signal till klockans mottagare, visade ändå klockans display att dess mottagare fick en stark och tydlig signal från kretsen.

7 Beskrivning av signalens väg

För att lättare kunna analysera och beskriva signalens väg genom de involverade komponenterna beskrivs den här nedan i en förenklad tolkning.



Figur: 6 Beskrivning av signalens väg mellan de inblandade komponenterna

Det hela börjar med att EXO3-komponenten(kristalloscillatorn) sänder en kontinuerlig frekvens på 20 MHz till DCF77-komponenten och en kontinuerlig frekvens på 10 MHz till processorn.
Vad gör då dessa frekvenser?

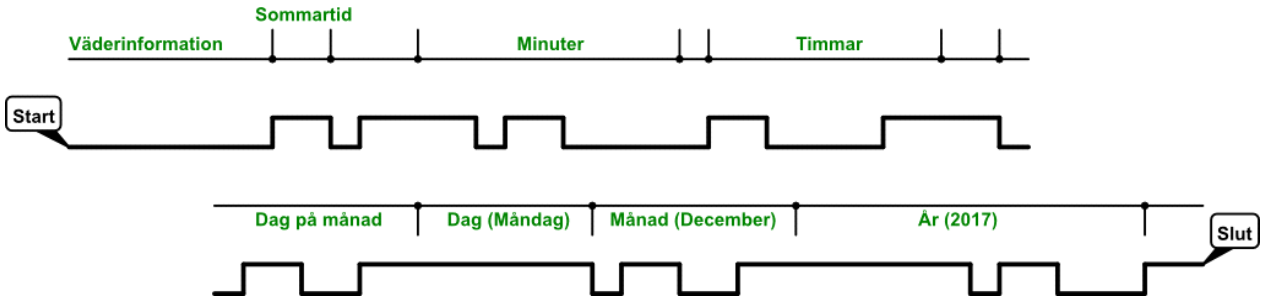
- Signalen på 10 MHz går till processorn och används som en extern klockpuls, vilket är den hastighet som processorn kan utföra instruktioner.
- Signalen på 20 MHz används som grundfrekvens till DCF77-komponenten för vidare behandling. Se kapitel 6.5 för vidare förklaring av hur denna signal behandlas.

Efter att DCF77-komponenten behandlat denna ingående signal på 20 MHz skickas dess utgående signaler i form av frekvenser dels till processorn och till modulatorn(se figur 6).
Vad dessa frekvenser gör är följande:

- Dessa utgående frekvenser, den första på 10 Hz används som avbrottssignal till processorn, vilket ger väldigt exakta avbrott.
- Den andra utgående frekvensen är på 77,5 kHz går till modulatorn och är den korrekta frekvensen som modulatorn kräver för att så småningom skicka en bra antennsignal enligt DCF77-standard.

Som sista steg i signalens väg är modulatorn. Denna komponent får dels ovannämnda grundfrekvens på 77,5 kHz samt en styrsignal från processorn. Denna styrsignal från processorn är de bitmönster som representerar den tidsangivelse som användaren matat in. Styrsignalen avgör om modulatorn skickar ut en hög eller låg signal till antennen.

När den färdiga signalen skickas till antennen och så småningom fångas upp av mottagaren startar klockan en mätserie på denna signal. Det är klockans interna hårdvara och mjukvara som bearbetar den uppfångade signalen och kontrollerar dess trovärdighet.
På grund av denna inbyggda kontroll krävs det att den signal som skickas är korrekt i alla avseenden under varje sekund.



Figur: 7 Modell av en delsändning

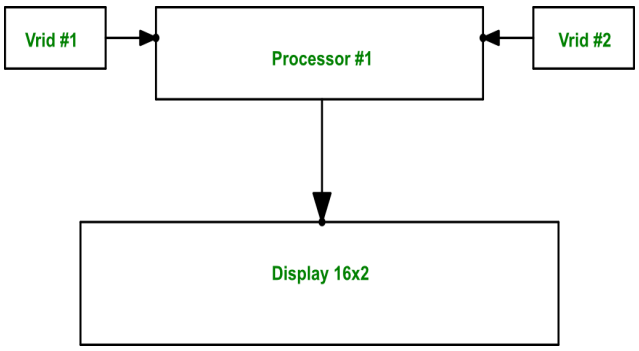
8 Genomförande och tillvägagångssätt vid konstruktion av projektet

Innan gruppen fick klartecken från handledaren att starta med den verkliga planeringen och genomförandet av projektet var gruppen tvungen att visa upp en grovplanering och skiss över hur projektet skulle utföras. I denna angavs vilka resurser som skulle komma att krävas från både gruppens och handledarens sida för att projektet skulle lyckas. Då gruppen fått klartecken från handledaren inleddes projektet med ett möte där en mer detaljerad planering gjordes. Delmål sattes upp för att tydliggöra framstegen i projektet samt underlätta eventuell felsökning. Detta utgjorde en viktig del i gruppens planering då dessa kontrollpunkter kom att tjäna som avstämningspunkter för vad som återstod att göra.

Tidigt i projektet stod det klart att det krävdes en ganska omfattande studie av datablad innan konstruktionen kunde påbörjas. Datablad rörande i stort sätt alla komponenter som används i projektet. Denna noggranna studie av datablad gjordes för att försöka minimera antalet misstag vilka skulle vara resultatet av att för lite tid lagts på förberedelser.

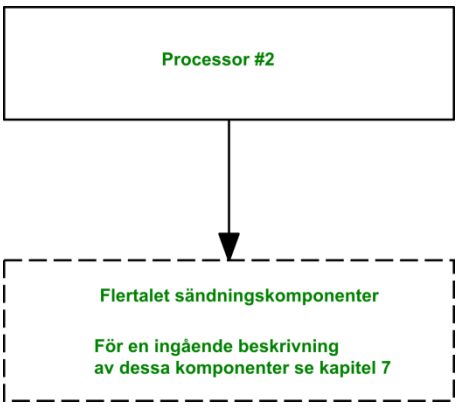
En uppdelning inom gruppen genomfördes varvid det bestämdes att varje subgrupp om två personer fick var sin processor att arbeta med. Inledningsvis bestämdes det att den ena gruppen skulle programmera en processor som kommunicerarmed displayen och vridreglagen, den andra delen av gruppen fick i uppgift programmera processorn som hanterar sändningen.

Projektets uppdelning visas som 2 block i figur 8 och 9. Det första blocket är följande.



Figur: 8 Block 1

Processor 1 skall kunna kommunicera med 2 vridreglage som tillsammans bildar den del som låter användaren göra inställningar i form av datum och tidsinställning. Inställningarna visas i realtid på displayen som kommunicerar med processorn.

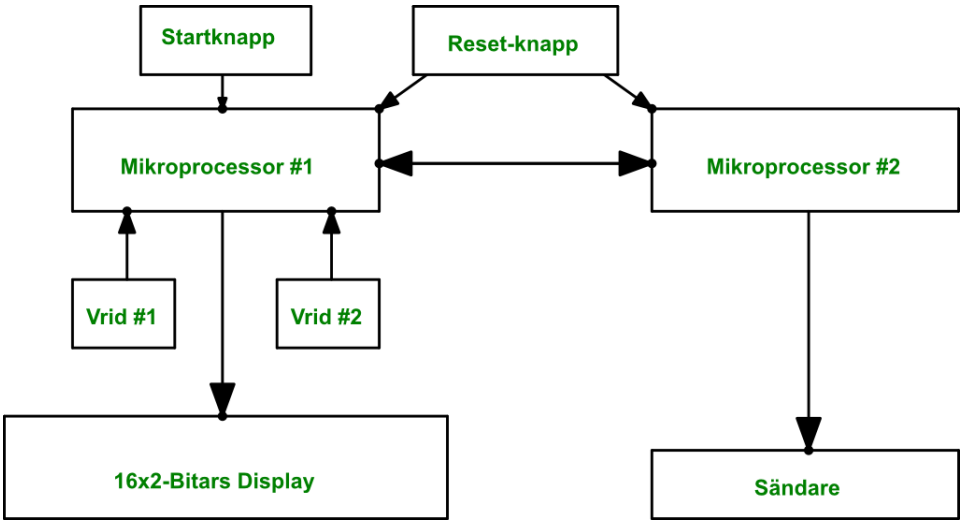


Det andra blocket är följande.

Processor 2 skall kommunicera med de komponenter som kommer att behandla sändningen. När processor 2 har tagit emot de data som användaren ställt in i ”block1”.
Denna information som tagits emot skickas genom en vidare och en serie av sammankopplade komponenter krävs för att signalen skall bli korrekt. Denna process beskrivs i kapitel 7.

Figur: 9 Block 2

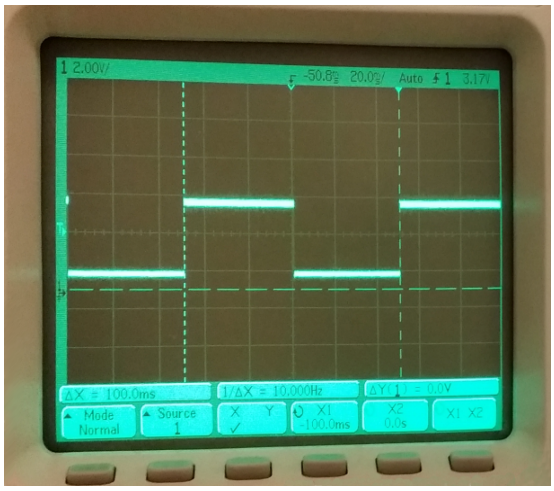
En sammanslagning av de 2 block som illustreras i figur 8 och 9 visas här nedan i figur 10.



Figur: 10 Det första blockschemat som upprättades av konstruktionen

Något som fort kom att visa sig var att fler komponenter krävdes för att bearbeta och modulera den signal somskulle skickas mellan de olika komponenterna för att bibehålla korrekt frekvens. Dessa komponenter finns beskrivna i kapitel 7.

Under tilden som arbetet med signalbehandlingen fortskred gjorde arbetet med displayen framsteg. Enskillda tecken och strängar kunde nu skrivas till displayen. Det fortssatta arbetet bestod nu av att kunna lagra textsträngar och hämta textsträngar vilka represneterar menyn från EEPROM-minnet. Ett särskillt skrivningsprogram har framtagits för att underlätta skrivning till EEPROM-minnet då detta minne lätt kan bli korrupt vid programskrivning. Dessa textsträngar skall vid start av apparaten hämtas från EEPROM-minnet.



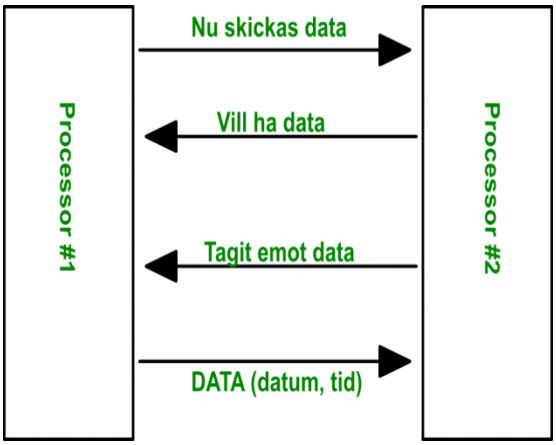
Innan några större kopplingsmässiga framsteg kunde göras kvarstod mycket läsande av datablad för de olika komponenterna.

Då arbetet med sändningsrutinen fortskred gjordes ständiga mätningar för att se till att allt fungerade som det var tänkt att göra. Logikanalysatorn var under detta skede en ovärderlig tillgång. I figur 11 kontrolleras den signal som används som avbrottssignal till processor 2 med urspurung från CPLD-komponenten. En tydlig fyrkantsvåg kan utläsas på logikanalysatorns skärm där amplitud och period överänsstämmer med de värden angivna i databladet.

Figur: 11 Kontrollmätning

Under tiden som kretsen fick fler komponenter och nya kopplingar gjordes dememellan, stod det klart att ett ordentligt kopplingsschema över vilka sladdar som var dragna till vilken komponent och vilket bennummer skulle komma att behövas. Ett sådant schema togs fram för att underlätta det fortsatta arbetet.

Då de båda blocken var relativt klara, med det menat att allt i teori skulle fungera, påbörjades arbetet med att försöka sammankoppla dessa block för att fungera som en gemensam enhet.



Det sista steget i konstruktionen var själva sammankopplingen, vilken länkar samman de båda processorerna och tillåter dem att fungera som en gemensam enhet.
För att veta när processor 1 skall skicka data respektive när processor 2 skall ta emot data används en så kallad handskakning, vilket består av ett antal kontrollbitar som indikerar när data skickas samt när data har skickats. Detta lilla protokoll över denna handskakning illustreas i figur 12.

Figur: 12 Överföringen mellan de båda processorerna i en grafisk tolkning

9 Konstruktion av mjukvara

Mjukvaran är programmerad i assembler för atmega8 och utvecklad i AVR-studio.

Mjukvara för processor 1

Bit: En bit är en datamängd som antingen kan vara ett eller noll, d.v.s. att den är binär.
I Atmega8-processorn är det mesta 8 bitar stort, detta betyder att bitarna är hör ihop för att kunna representera större värden. En rad bitar läses från höger till vänster och varje bit representerar ett värde som är dubbelt så stor som biten till höger om den, med undantag för den längst till höger som representerar ett. När en bit är en etta så adderas värdet den representerar till den totala summan.

Port: En port som används för att kunna kommunicera med processorn, dessa går att ställa in för att skicka data antingen ut eller in, eller delvis ut och in samtidigt. På Atmega8-processorn så finns port B, port C och port D.

Register: En plats att lagra värden som används och ändras ofta, på Atmega8-processorn finns det 32 st av dessa, alla 8 bitar stora.

EEPROM: ElectricallyErasableProgrammable Read-OnlyMemory, programmerbart minne som är långsamt, men användbart för data som väldigt sällan ska ändras.

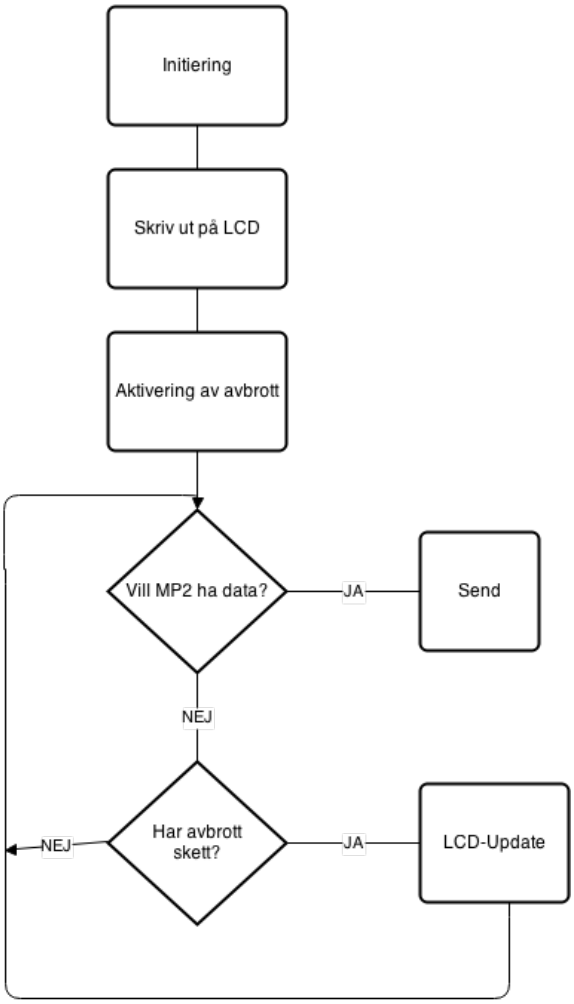
Talbas: Det finns ett antal olika talbaser som används på grund av att deras fördelar inom vissa områden. Den talbas som används i folkmun som alla är vana vid kallas för decimal. Processorn lagrar värden hexadecimalt, detta betyder att värden går från 0 till F (0 till 9, A till F), där F står för decimalt 15.

Stackpekare: En del av minnet som används för att kunna spara undan vart den nuvarande exekveringen sker, för att på så sätt kunna hoppa till andra platser i koden för att senare kunna återgå till denna plats.

Avbrott: Ibland behövs en del kod som (oftast) ska kunna exekveras när som helst då ett avbrott sker. Då kommer processorn spara undan dess position i stackpekaren, och sedan börja exekvera avbrottskoden.

ASCII: American Standard Code for Information Interchange är en tabell som används för att kunna använda värden till att representera olika siffror och tecken.

Huvudprogram
Efter initiering av stackpekare och displayer uppdateras displayen med uppstartsvärden, efter detta aktiveras även avbrott. Sedan sätts programmet i en loop, där den först kontrollerar om processor 2 vill ha data genom att titta på en specifik bit. Om inte så är fallet tittar den om avbrott har skett. Om ingen av dessa uppfylls upprepar den samma procedur. Upptäcker programmet att processor 2 vill ha data hoppar den till sändningen. Ifall ett avbrott har skett uppdaterar den displayen med korrekt information.



Figur: 13 Flödesschema över huvudprogram för processor 1

Avbrott

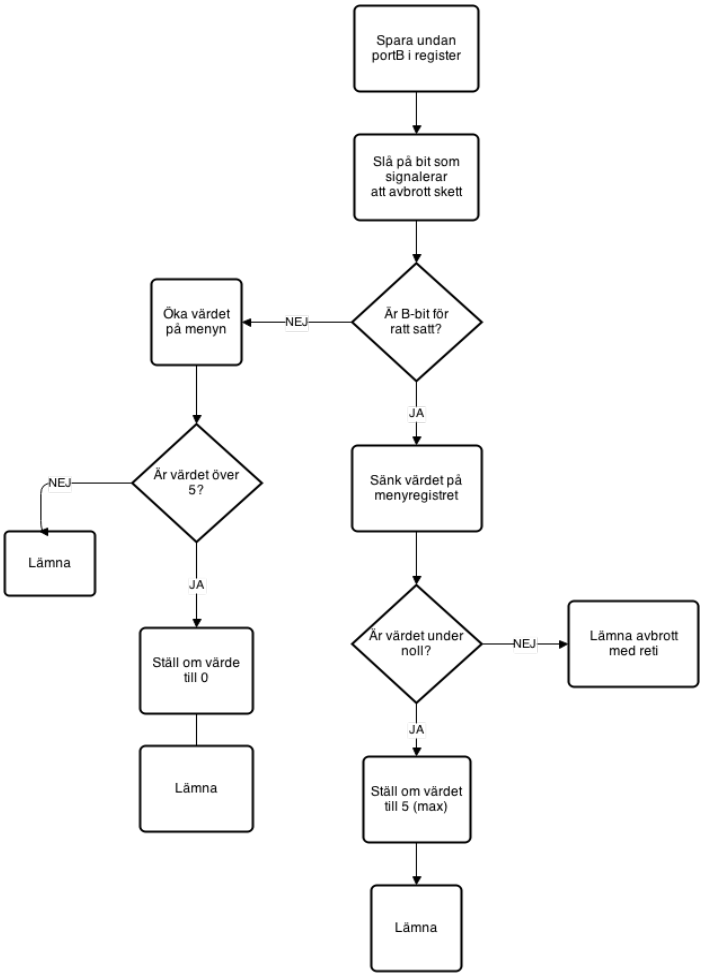
I programmet används två avbrott, det ena avbrottet (Avbrott 1) används för att byta vilken meny användaren är på (år, månad, dag, veckodag, timme, minut), och den andra (Avbrott 2)används för att ställa in vad den delen skall ha för värde. Båda avbrotten läser av en bit i början för att kunna veta åt vilket håll vredet som aktiverade avbrottet har rört sig, denna bit kallas för B-bit i de flödesscheman som följer.

Avbrott 1

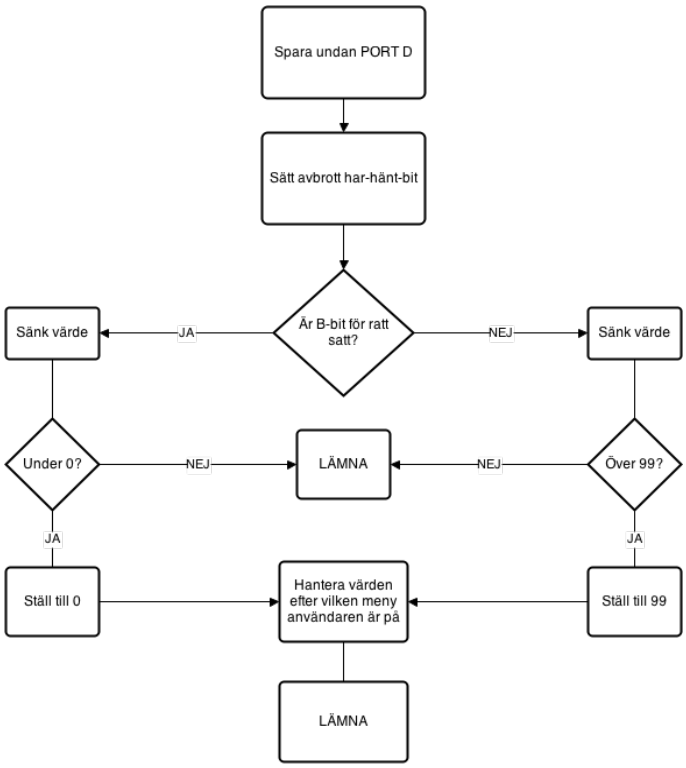
Det första avbrottet ökar eller sänker värdet som hanterar vilken meny man är på, då det är 6 st olika menyer tittar den om man ligger utanför 0-5

Avbrott 2

Avbrott 2 gör samma sak som avbrott 1, med undantaget att värdet är för den menyn användaren är på, samt att den går mellan 0-99.



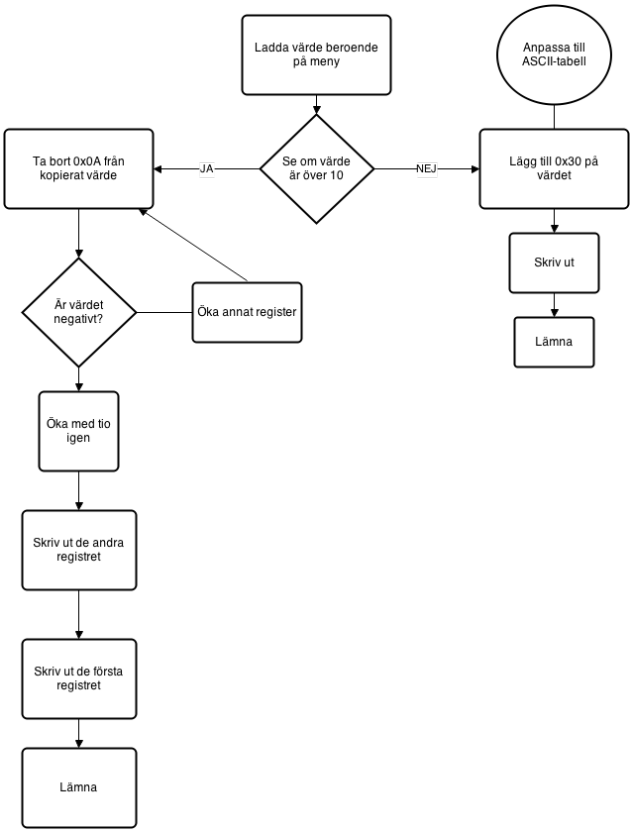
Figur: 14 Flödesschema avbrott 1



Figur: 15 Flödesschema avbrott 2

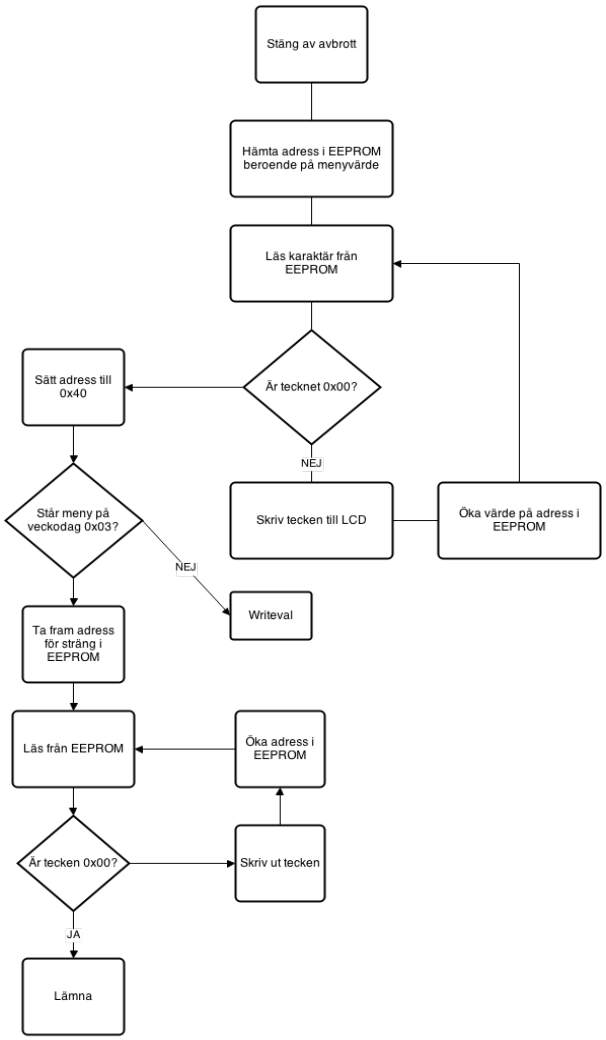
Uppdatering av displayen
Vid skrivning till displayen ska 8-bitar skickas, displayen tar då emot detta och tittar sedan i sitt minne efter värdet och skriver ut symbolen som finns där. Minnet följer en del av ASCII-tabellen om denna inte avsiktligt har ändrats. När displayen skall uppdateras rensas den först. Sedan skriver den ut en sträng på övre raden visa för att vilken meny användarenär på.Därefter skiftar den till undre raden (adress 0x40) och skriver ut värde eller sträng. Dettaberoende på vilken meny användaren är på, som visas i figur 3.

Utskrift av hexadecimala värden
För att kunna skriva ut hexadecimala tal i en decimal talbas avgörs först om talet är över tio, om vilket var sant betyder att det inte går att skriva ut normalt. För dessa fall avgör programmet hur stort tiotalet är genom enkel subtraktion och sedan skriver den ut både tiotalet och entalet som blir över.



Figur: 16 Flödesschema, utskrift av hexadecimala värden

Utskrift av sträng
Först och främst bestäms vilken adress i EEPROM som programmet skall börja på. Detta med hjälp av värdet på menyregistret. Efter detta läser den ett tecken och undersöker om tecknet är 0x00, vilket representerade radslut, om inte så skriver den ut och stegar upp adressen i EEPROM.

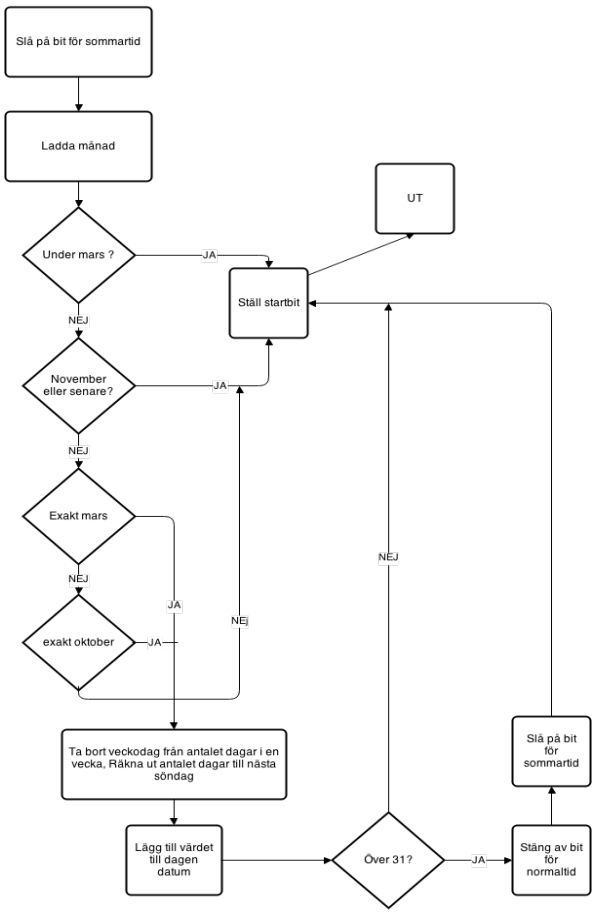


Figur: 17 Flödesschema, utskrift av strängar

Sändning
Sändningen sker i två steg, först konverteras alla lagrade värden för att överensstämma med DCF77-kodningen, efter detta skickas alla dessa till den andra processorn.

Konvertering
I början tar programmet reda på om sommartid gäller eller inte, här tittar den först om användarenhar valt en månad där sommartid inte växlar, vilket gör att man lätt vet om sommartid ska gälla eller inte. Om en månad väljs därsommartid växlar räknar programmet ut vilket datum nästa söndag ska vara för månaden. (Om en söndag valdes som veckodag ökar den inte.) Om då detta datum har gått utanför månadens längd har sista söndagen i månandenpasserats och sommartid baseras på vilken månad som användaren valt.
Efter detta börjar programmet konvertera värdena, och då DCF77-kodningen är väldigt lik den decimala talbasen kunde koden för hexadecimal utskrift återanvändas, dock med små modifikationer för att hantera vissa undantag

Sändning
Sändningen skickas bit för bit, med extra bitar som processorerna använder för att hålla reda på när nya värden skall skickas eller läsas. Detta för att säkerställa att gamla värden inte läses in dubbelt, då processor 2 hade betydligt högre klockhastighet än processor 1.

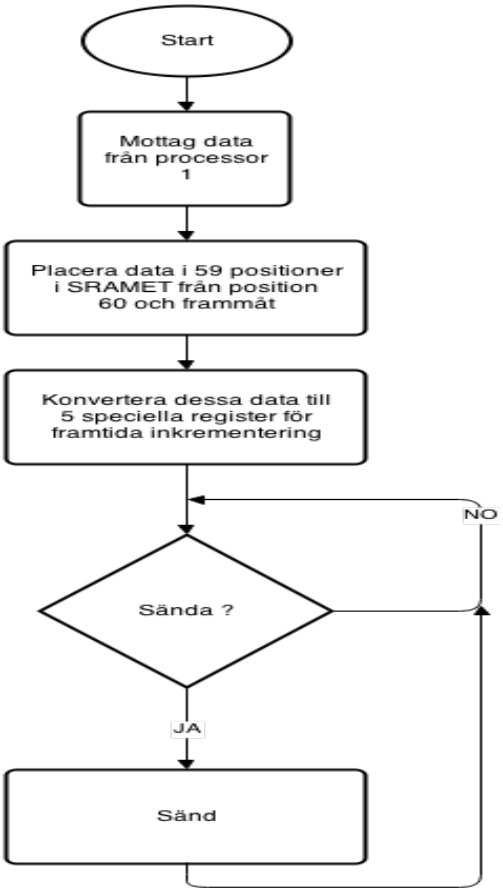


Figur: 18 Flödesschema, sändningsrutin

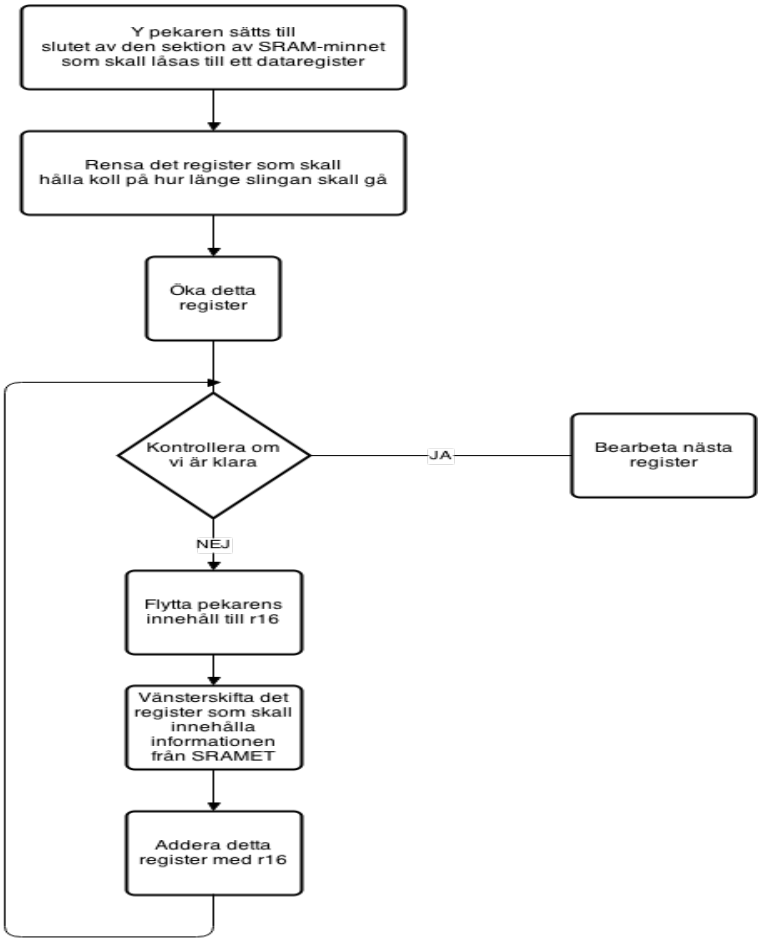
10 Mjukvara för processor 2

Mjukvaran för processor två är uppdelad i fyra huvudsakliga funktioner. Inledningsvis är en initialiseringsrutin programmerad för att sköta överföringen från den första processorn och därefter uppdatera SRAM-minnet med informationen som skall sändas ut. Därefter följer det en rutin som tar de 60 bytes nu placerade i SRAM-minnet och konverterar dessa till 6 dataregister, vilket komprimerar tidsinformationen som behandlar minuter, timmar, dagar, veckodagar, månader samt år bestående av 60 bytes till 6 bytes stora dataregister, se figur P2.2. Detta för att senare inkrementera dessa register så att den tid som sändningsrutinen skickar ut uppdateras precis som en vanlig klocka.

När den inledande mjukvaran exekverats lägger sig programmet i en vänteslinga där processorn väntar på kontinuerliga externa avbrott på 10 Hz. Figuren nedan visar den huvudsakliga funktionaliteten hos processor 2 d.v.s. att operera den externa antennen genom att vid sändning sänka spänningen och på så sätt åstadkomma den amplitudmodulering som är nödvändig för att skicka ut DCF77-signalen.



Figur: 19 Flödeschema över den generella funktionaliteten

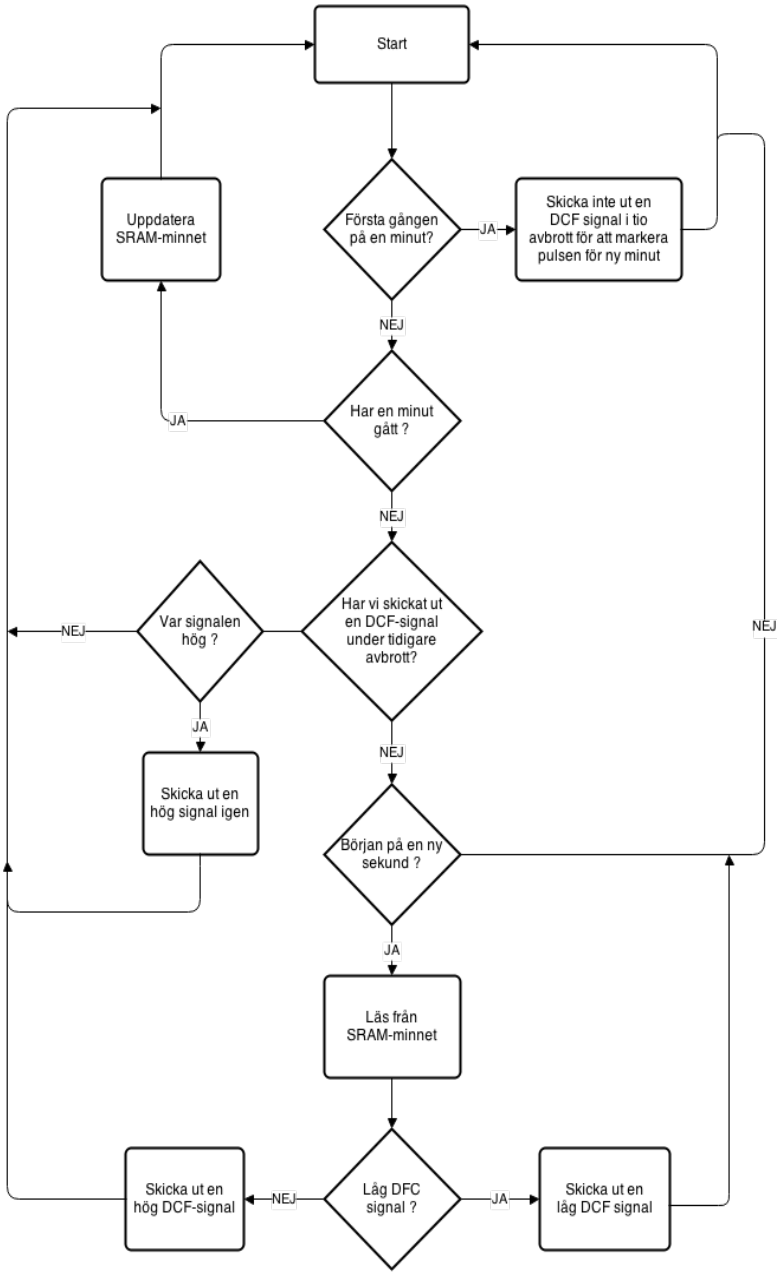


Figur: 20 Konverteringen från SRAM till register

Avbrottsrutinen

Avbrottsrutinen figur P2.3 hanterar amplitudmoduleringen av DCF77-signalen på 77,5 kHz genom att den beroende på innehållet i SRAM-minnet sänker spänningen i modulatorn. Detta gör den genom att skicka ut en låg signal i 100 ms om en logisknolla skall sändas eller i 200 ms om en logisketta skall skickas ut till den externa antennen. Rutinen gör detta genom att varje sekund flytta en pekare mellan totalt 59 minnesplatser och behandla de data som ligger däruti.

I början på varje ny minut kommer dock inte avbrottsrutinen göra någon amplitudmodulering utan avbrottsrutinen kommer att gå tom i 10 avbrott; d.v.s. tidsmässigt en sekund. Detta för att det enligt standarden för DCF77 kräveren kort paus under en sekund i början av varje minut. För att signalera till mottagaren att en ny minut skall påbörjas.

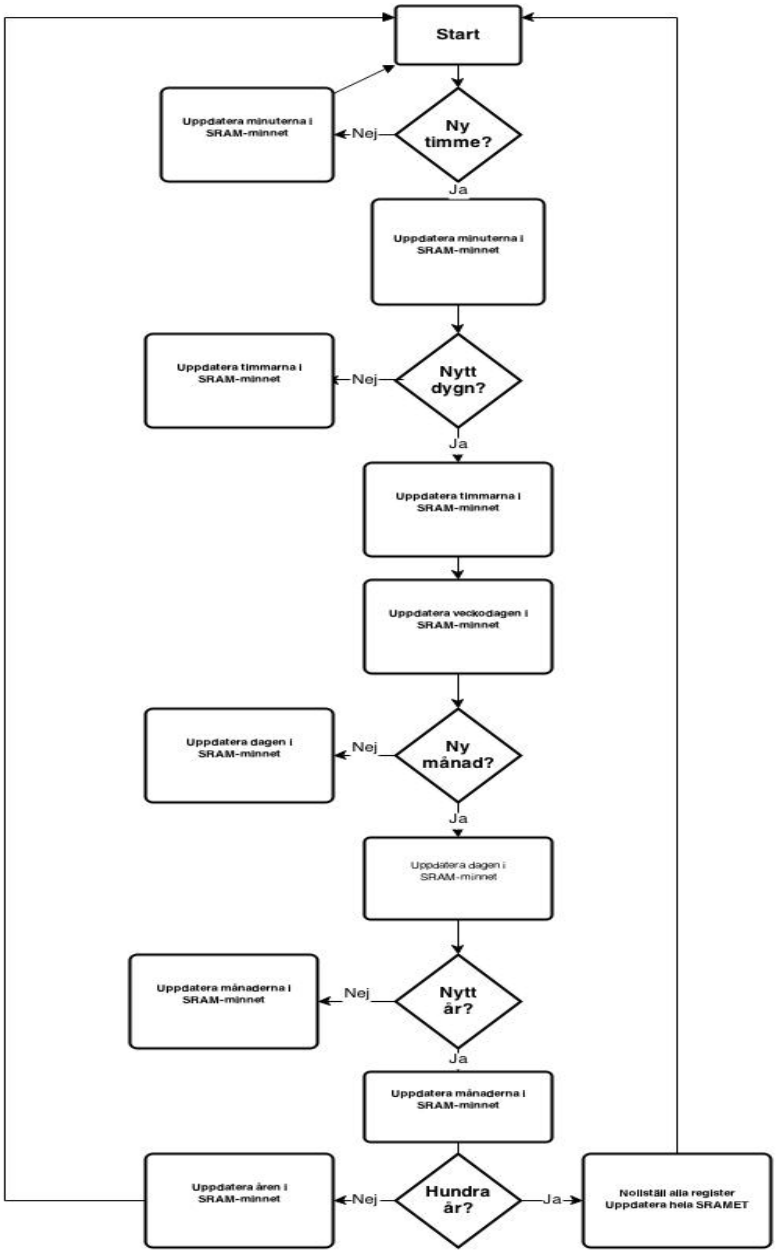


Figur: 21 Flödeschema över DCF77-sändingen

Tidsinkrementering

För att hålla koll på tiden har avbrottsrutinen en sekundräknare som kontrollerar om en minut har passerat. När en minut väl förflutit anropas subrutinen som uppdaterar de register som används för att hålla koll på kalenderdatumet och nuvarande klockslag. Subrutinen kommer i normalfallet att inkrementera de dataregister som hanterar minuterna och sedan anropa ytterligare en rutin som skriver registerinnehållet på 1 byte till 7 bytes i ramminnet, men i vissa fall kommer månader eller år uppdateras parallelt. Dessa fall måste hanteras och dessutom måste paritetsbitarna vara korrekta därför kommer den senare subrutinen att kalla på ytterligare en subrutin föra att sätta rätt paritet.

Detta då det enligt DCF77-standarden fodras totalt 3 paritetsbitar. En paritet för minutpulserna, en paritet för pulserna som hanterar timmarna och slutligen en paritet över dagarna, veckodagarna månaderna och åren. Se tabellen 2.



Figur: 22 Flödeschema över kalendern i P2

När en timme passerat fortsätter programmet med samma beteende som ovan. Fast för registret innehållandes timmar d.v.s. den kommer att uppdatera innehållet i SRAM-minnet som hanterar timmarna med paritet på samma sätt som den behandlade minuterna.

Denna del av mjukvaran för processor 2 fungerar som en kalender se figur 22 till höger. Detta för att kontinuerligt emulera den korrekta sändningen i Mainflingen. Programmet kommer alltså kontrollera vilket datum vi har och justera dagarna och signalerna efter den givna tid som erhålles från processor 1. Detta då de klockor som hanterar DCF77 signalen allt som oftast har en inbyggd kalender, dessa klockor erfordrar därför en kalender som stämmer överens med den inbyggda.

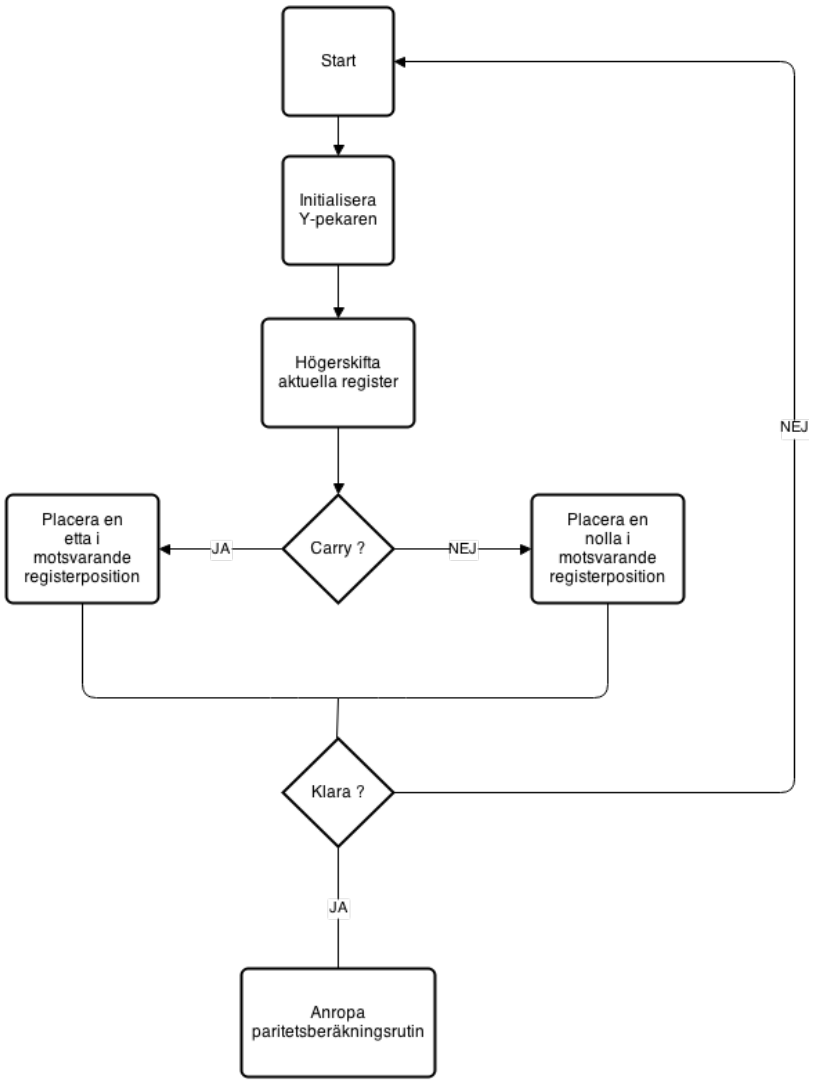
Skrivning till SRAM-minnet

Figur 23 visar algortimen som skriver till SRAM-minnet. Skrivningen inledes med en initialisering av Y-pekaren. Därefter följer ett högerskifte av det register som förtilfället skall skrivas till SRAM-minnet. Skulle en carry erhållas vid högerskiftet placeras en etta i motsvarande registerposistion och ifall högerskiftet inte skulle resultera i en carry kommer en nolla att skrivas till motsvarande position i SRAM-minnet.

Denna procedur pågår tills rutinen är klara och fungerar i stort sätt likadant för alla register som hör till kalendern. Vilka också har egna upplagor av samma rutin. Skillnaden ligger främst i vilken position i SRAM-minnet rutinen skriver till.

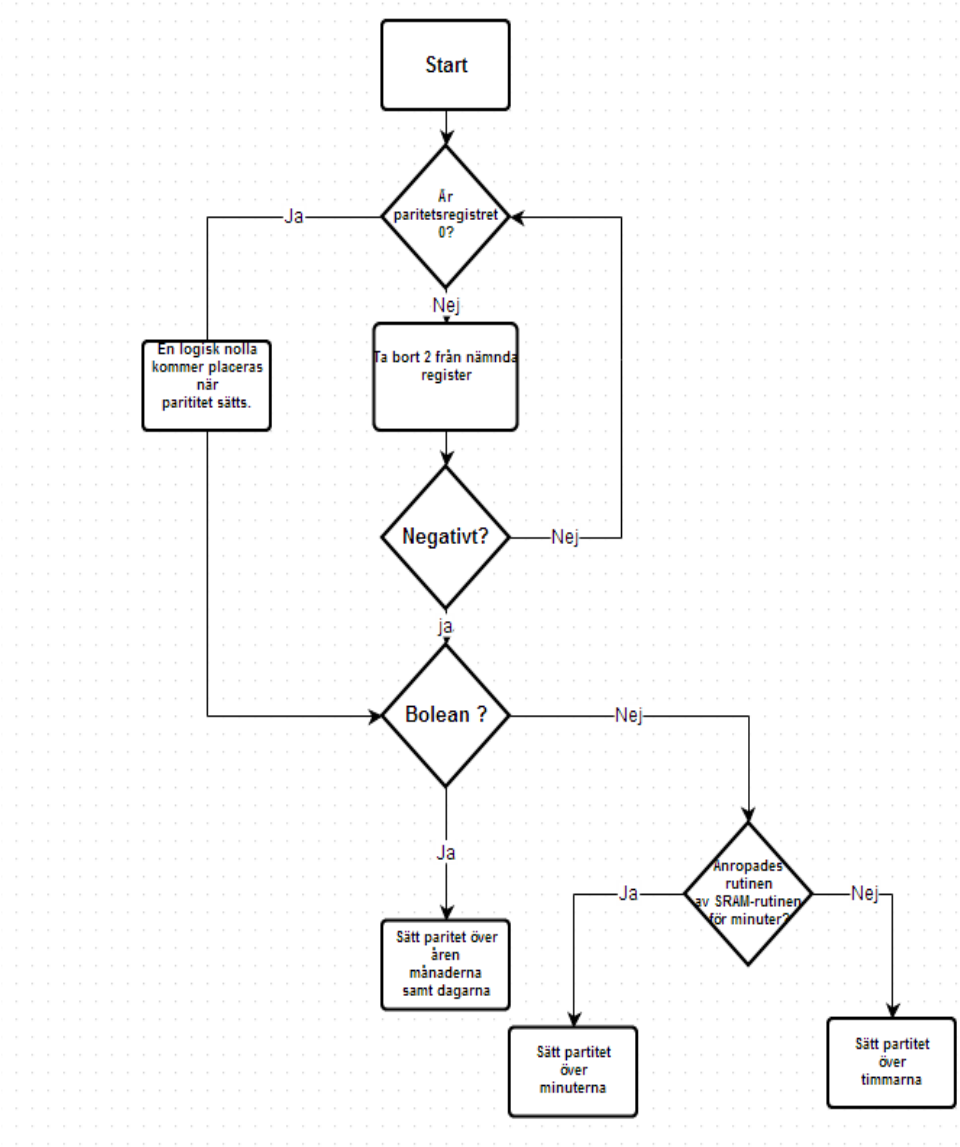
När en etta skrivs till en position till minnet kommer emellertid ett annat register uppdateras. Nämligen det register som används för paritetsbitsberäkningarna.

Detta för att senare kunna sätta rätt partitet i paritetberäkningsrutinen.



Figur: 23Flödesschema över skrivning till SRAM-minnet

Partitetsberäkningsrutinen.
Figuren till höger visar paritetsberäkningsrutinen. Paritetsberäkningsrutinen för processor två fungerar i huvudsak på samma sätt som motsvarande rutin i processor 1. Den viktiga skillnaden är att processor2 räknar upp resultatet och skriver till SRAM-minnet därför är ett boolskt utryck implementerad så att samma paritetsberäkningsrutin skall kunna användas för alla subrutiner som skriver till SRAM-minnet.



Figur: 24 Visar paritetsberäkningsrutinen för processor2

11 Problem och svårigheter som uppstått under projektets gång

Ständigt under projektets gång har nya utmaningar uppkommit av olika svårighetsgrad, men som alla på något sätt behövde lösas. Problem som uppkommit har dels varit hårdvarurelateradedå en del sladdar kopplats fel på grund av feltolkning av datablad. Andra fel har varit syntaxmässiga fel i koden och fel som uppkommit då olika delar av projektet skulle sättas samman.

Det största fel som uppkom uppstod då två separata delar av projektet skulle sammankopplas. Felet som inte syntes då delarna fortfarande var två separata enheter blev ett resultat av sammanlänkningen mellan dem. Det tog en bra stund innan felsökningen ledde fram till att det endast kunde vara frågan om att avbrotten från de vridreglage som används vid menyn kolliderade med en annan rutin i koden vid en jämförelsesats. Det hela berodde på att z:a flaggan (en kontrollbit som sätts vid en jämfärelse) inte återställdes tidigare i koden vilket resulterade i detta mycket märkliga fel. Den slutsats som kan dras är att en ordentlig dokumentation, strukturerade kretsscheman och kontrollöver vilka flaggor som är satta respektive icke satta är av yttersta vikt.

12 Hur projektet kan utvecklas

Det finns många olika möjligheter för detta projekt att utvecklas och i vissa avseende förbättra konstruktionen och prestandan. Några förslag på förändringar och utökningar skulle exempelvis kunna vara att:

- Utöka den räckvidd som den nuvarande antennen har för att klara av att sända de radiofrekvenser som komponenterna nu genererar en betydligt längre sträcka,

- Byta ut den nuvarande displayen mot en större modell för att på så sätt kunna visa mer information ifrån båda processorerna samtidigt,till exempel vilken starttid som skickades, hur länge tiden har skickats och den nuvarande tiden som skickas.

13 Resultat

Av den kravspecifikation som upprättades innan projektets start kunde nästan alla punkter genomföras. Anledningen till att inte alla mål kunde genomföras beror på att det redan i ett tidigt skede av projektet kunde konstateras att den interna logiken hos de radioklockor projektet riktades emot inte var konstruerade att klara av detta krav. Det var alltså inte tekniskt möjligt att utföra.

- ☒ Enheten skall kunna agera ”föreläsningsklocka” dvs. skynda på minuterna 15–60och sakta ner minuterna 00–15.

Förövrigt har följande krav uppfyllts:

- ☑ Apparaten skall på ett säkert sätt kunna meddela tid till en radiokontrolleradklocka.
- ☑ Apparaten skall kunna sätta tid och datum helt godtyckligt.
- ☑ Apparatens räckvidd skall vara mindre än 20 cm.
- ☑ Klockans noggrannhet skall vara bättre än 60 sekunder per dygn.

14 Utvärdering av projektet

Gruppen har under projektets åtta veckor inte bara fått en god insikt i de tekniska komponenterna och hur de samverkar för att framstå som en gemensam enhet utan även fått goda erfarenheter kring hur mycket arbete som krävs för ett projekt av större storlek.

Något som gruppen har uppfattat som en svårighet under projektets gång är bristen på synliga framsteg mellan kod, hårdvara och den slutliga apparaten annat än de resultat som syntes på logikanalysatorn. Det var först mot slutet av projektets gång som en riktig uppkoppling kunde göras mellan hårdvara och mjukvara för att se om det skulle fungera.

Detta projekt har verkligen visat vikten av ett bra samarbete, kommunikation, dokumentation och ett väl strukturerade tillvägagångssätt vid felsökning av både hårdvara och mjukvara är ovärderligt i ett projekt som detta.

Projektet är mycket lärorikt då det knyter samman tidigare kunskaper och erfarenheter rörande hårdvara, programmering och teoretisk förståelse för elektriska lagar och kopplingar.