Om mikrodatorrapporten.

Det brukar frågas ganska mycket om hur rapporten egentligen ska se ut. Jag brukar hänga ut ett exempel i labbet när den tiden närmar sig. Avsikten är inte att framtida rapporter skall se likadana ut, avsikten är att ge en uppfattning om detaljeringsgraden som förväntas av en sådan rapport. Delar vi ut en "Standardrapport 1A" är risken tyvärr överhängande att alla rapporter ser likadana ut, vilket ju är helt orimligt då projekten, och framförallt författarna, skiljer sig åt.

Det du läser nu är en diskussion och liten beskrivning av vad man kan tänka på när man skriver sin rapport.

Tänk på vem läsaren är. För Mikrodatorprojektets skull tänker vi oss att rapporten skall skrivas till person som kanske läst digital- och datorteknik men inte mer. Rapporten behöver inte beskriva *allt*, bara hela projektet.

Rapporten skall dokumentera projektet. Som författare måste du se till att rapporten inte bara är korrekt utan också är *läsvärd*. Det räcker inte med att den innehåller all information, den måste också presentera informationen på ett sådant sätt att den kan läsas och, helst enkelt, förstås av läsaren!

Det går inte att undvika ett visst mått av undervisning förutom själva faktainnehållet.

Gör allt du kan för att underlätta för läsaren. Beskriv förutsättningarna, beskriv komponenterna, beskriv tillvägagångssättet, beskriv hårdvarukopplingarna, beskriv mjukvaran, beskriv handhavandet osv.

Här nedan kommer lite lösa exempel att studera. Läs och försök dra slutsatser angående *avsikten* och *syftet* med rapporterna. Använd sedan dessa slutsatser för din egen rapport.

Om du som författare inte lyckas överföra din information är det inte läsarens fel, det är ditt!

Exempel 1: 10 W Br m/39

Som exempel på hur en beskrivning kan göras använder jag först exemplet med den militära radiostationen 10 W Br m/39, hädanefter kallad *beskrivningen*. Den beskrivningen går att använda då den är upplagd på ett sätt som fortfarande är aktuellt och samtidigt är den innehållsmässigt rätt långt från våra mikrodatorprojekt. Avsikten är så klart inte att din grupps rapport skall vara identisk med radiobeskrivningen men flera goda tankar kan lånas från den.

- Hela rapporten måste vara strukturerad för att kunna läsas enkelt. Bland det första intrycken man kan ge läsaren är därför att innehållsförteckningen är vettig. Här avslöjas innehållet med hjälp av valda *kapitelrubriker* och *indentering*.
- Tänk hela tiden på att beskriva först översiktligt och sedan mer och mer i detalj. Redan ur innehållsförteckningen kan man se att författaren strukturerat framställningen:

  4 Funktion

  Koncentrerad beskrivning

  Sändaren

  Mottagaren

  Detaljerad beskrivning

  Sändaren

  Styroscillatorn

  Effektsteget

  :

  Redan här har man fått en del nyckelord om innehållet och är man bekant med dessa kan man avgöra om man vill läsa vidare eller inte. Det vore ju tråkigt att behöva läsa igenom hela rapporten för att konstatera att den inte innehöll något intressant. Du som rapportförfattare måste göra presentationsjobbet så att innehållet framgår.
- Att man går från översiktlig nivå till en mer detaljerad nivå framgår av resten av beskrivningen. Speciellt ska man tänka på att inte diskutera detaljer som läsaren inte fått presenterade för sig.[1]
- Bilder och figurer är starka förmedlare av information. Använd bilder för att få med läsaren i beskrivningen. Se till att ha åtminstone tre beskrivande textrader till varje bild. Hur läser du själv en bok med bilder och bildtexter i? Jo, genom att titta på bilderna och läsa biltexterna såklart. Efter detta kommer "den stora genomläsningen".

[1] Inled alltså inte hela dokumentet med "Vi gjorde en skywriter. Mellan pinne 3 på LCD:n och plus satte vi potentiometern." Det blir fel på så många plan: Som läsare vet man inte här vad en skywriter är, än mindre att en LCD behövdes (vet man säkert vad det är för något heller?) och ännu mindre vad pinne 3 är och varför den behöver en potentiometer.

Exempel 2: SNAKE-projekt

Framsidan ger första intrycket:

SNAKE

Mikrodataprojekt - grupp 6:s

Axel Blackert, Oskar Eriksson, Oscar Fredriksson, Filip Herslöf

VT1 2014

Denna rapport innehåller en detaljerad dokumentering av Grupp 6 konstruktion av Snake. Rapporten täcker hur komponenter kopplas och hur de samspelar med en Atmel-miljö för att sedan fungera som ett gammaldags Snake-spel.

Här ser man vad det är för projekt, till och med hur det ser ut *och* en beskrivande text på vad det är för något man har framför ögonen alternativt håller i handen om man skriver ut den. Allt solklart för läsaren!

Kravspecifikation Kravspecifikationen som bestämdes vid projektstart ligger som underlag för projektdokumentationen. För att förankra projektets omfattning hos läsaren måste den redovisas:

1.2 Kravspecifikation

I början av projektet utgick vi från en planering som innehöll våra mål för projektet. Dessa mål var som följer:

- Ormen ska röra sig framåt med konstant hastighet, bli längre vid kollision med ”mat” och spelet ska avslutas vid kollision med sig själv eller kanterna (slutet av spelplanen, illustrerar att spelaren inte kan gå utanför spelplanen) av planen.
- Spelet ska kunna spelas tills spelaren förlorar eller vinner, då ska spelarens poäng visas och sedan bli frågad om denne vill spela igen eller avbryta spelet.
- Spelet ska innan start fråga om svårighetsgrad (3 stycken) och dessa ändrar bland annat ormens hastighet.

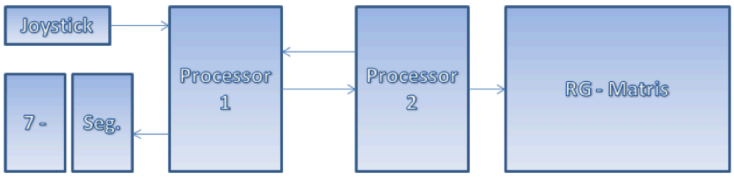
Under projektets gång gjorde vi dock några smärre ändringar på dessa krav. Dessa ändringar samt dess anledningar listas här under:

1. Kollision med kanterna avslutar inte längre spelet utan får endast ormen att komma ut på motstående sida. Vi gjorde denna ändring för att planen kändes för liten för att stänga in spelaren inom spelplanen.
2. Då spelaren förlorar visas en deathscreen och sedan stannar spelet. Vill spelaren spela igen så får denne starta om spelet med reset-knappen. Spelarens poäng visas dock både under spelets gång och efter avslutat spel.
3. Antalet svårighetsgrader ändrades från 3 stycken till 2 stycken. Anledningen för detta var endast för att startskärmen som låter spelaren välja svårighetsgrad blev snyggare med endast 2 alternativ. Dessutom verkade det onödigt med 3 stycken

Här har gruppen specificerat detaljerna och även redovisat några ändringar (smärre, enligt dem själva) som godkänts av kursledningen.

Det kan inte nog betonas hur viktigt det är att få med läsaren genom dokumentet. Gör det lätt för läsaren. Eftersom en stor del av hjärnan används för att behandla synintryck är också bilder och figurer viktiga och bra för att beskriva vad som händer:

Här följer en bild över hur hårdvaran spelar samman:



Figur: Flödesschema-Hårdvara.
Flödesschema för den hårdvaran som är inkluderad i projektet. Pilarna illustrerar signaler och visar i vilken riktning signalen går. Nedan följer detaljerad genomgång av schemat.

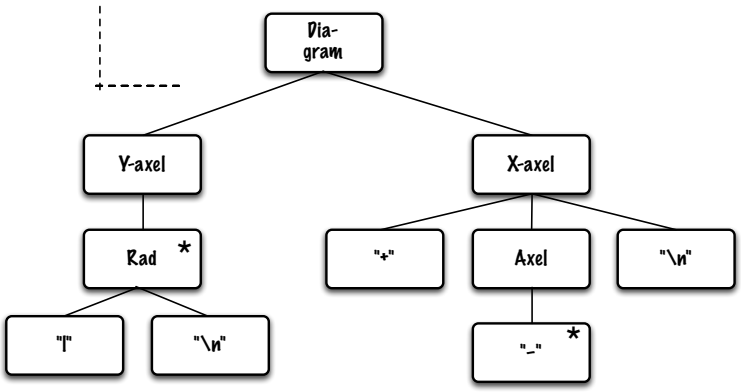
Här har författaren

1. föreberett läsaren på vad det är som kommer i figuren,
2. tillverkat en tydlig bild med inte för mycket klutter utöver den väsentliga informationen,
3. i figurtexten angivit vad figuren föreställer,
4. i figurtexten *beskrivit* vad figuren föreställer och hur man ska läsa den.

Utmärkt! En van läsare reagerar dock på ordvalet och stavningen i ”. . . over hur hårdvaran spelar samman” och att figuren saknar nummer. En revision senare skulle dessa mer kosmetiska effekter varit korrigerade.

Programdokumentation Den slutliga dokumentationen av kod är programvarulistningen. För att begripa denna behövs dock en beskrivning av programflöde, använda viktiga variabler, programkonventioner osv.

Det är översiktligt och beskrivande att programflödet redovisas i ett struktrudiagram enligt JSP:



(Här stoppade jag in ett generiskt JSP-diagram då vi hade inte börjat med det när detta dokument skrevs första gången)

För att begripa alla detaljer om vad som händer i diagrammet behövs mer text än vad som får plats i en figurtext. Figurtexten behövs dock fortfarande. Beskriv programmet också i löpande text:

4 Mjukvara – Master

Den av våra processorer som vi kallar för *masterprocessorn*, har tre huvuduppgifter.

1. A/D-omvandling
2. SPI
3. Poängutskrift på 7-segment

Processorn ”jobbar” i samma ordning som punkterna ovan . Först i huvudloopen börjar den med att läsa in X- och Y-värden från joysticken, skickar vidare värden med SPI och sedan skriver ut poängen på 7-segmenten. Denna processor är den processor med mindre rader kod, och vi har endast behövt använda 3 register.

```
.def SCORE = r17
.def TMP = r16
.def S_INPUT = r18
```

Man kan få en uppfattning om den önskade detaljeringsgraden i programdokumentationen från dokumentets innehållsförteckning:

Kommenterad, begriplig och läsvärd kod skall bifogas projektrapporten.

Hårdvaruschema/kretsschema

Beskriv ingående komponenter så att den fortsatta framställningen blir begriplig. Detta betyder troligen att processorn får en egen rubrik och text som beskriver processorn i allmänhet och hur den använts i projektet i synnerhet. Man kan nämna att processorn innehåller EEPROM/ADC/Timer och så vidare. Man ska definitivt ta upp de delar som använts ”För att kommunicera med omvärlden är processorn utrustad med hårdvara för olika seriella överföringar exempelvis USART, TWI och SPI. I projektet användes TWI.”

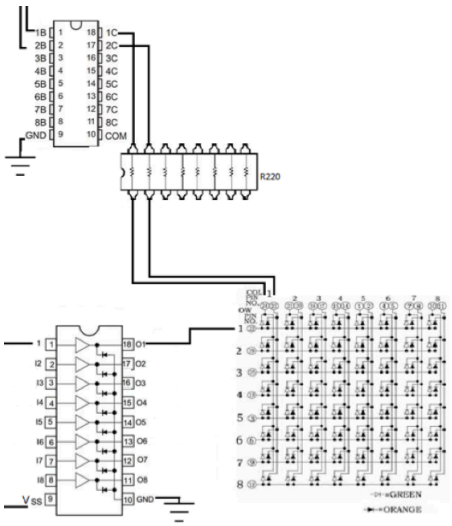
Under egen rubrik redovisar man sedan vad TWI är för något. Man behöver då inte ta upp *alla* egenskaper hos TWI men de som använts måste vara med. Rimligen beskrivs begreppen *master* och *slave* och övrig relevant TWI-relaterad jargong.

Om gruppen är speciellt glad/bedrövad över någon detalj, beskriv detta. Läsaren kanske överväger att använda TWI eller SPI i sitt nästa projekt och vill ha reda på just detta?

Vissa grupper har ägnat, i mitt tycke, oproportionerligt mycket tid åt potentiometern och kontrastspänningen på LCD:n. Det är inte fel. Antagligen har just detta varit speciellt krångligt eller så vill man visa på var databladet inte var tydligt eller något annat. Vem är att döma? *Var det viktigt för gruppen är det viktigt att det finns med i rapporten.*

Lägger du som läsare märke till att de senaste fyra styckena var jobbiga att läsa? Varför var det så? (Tips: Visst skulle en figur underlättat eller hur?)

Exempel: Diodmatris En 8x8-pixels lysdiodmatris är en typiskt krånglig komponent. Förutom en beskrivning av komponenten i sig har författaren här också beskrivit den elektriska inkopplingen för att tända en enda lysdiod. Det är troligt att det ligger rätt mycket databladsläsning och funderingar innan detta blev resultatet. Och då ska det helt naturligt också beskrivas i rapporten:



Dokumentet fortsätter sedan i god pedagogisk anda med att beskriva hur man styr hårdvaran för att tända en hel rad och sedan godtyckliga punkter på matrisen. Kapitlet avslutas med att beskriva hur man infört ett koordinatsystem för att kunna addressera dioderna i vilket mönster som helst. Allt är beskrivet på ett sådant sätt att det är hart när omöjligt att *inte* förstå *exakt* hur utskriften fungerar. Mycket bra!

Begripligt kretsschema över hela konstruktionen skall bifogas projektrapporten. Rita schemat enligt de vanliga konventionerna.

Exempel 3: RFID-projekt

Ett exempel på en mer sammanhängande, men inte komplett, rapport återfinns efter denna text. Notera att det fortfarande kan finnas saker att ändra på i den återgivna versionen. Hitta gärna dessa och diskutera inom din grupp för att göra din grupps rapport bättre.

Sen då?

Man lär sig skriva genom att skriva *mycket*. Övning ger även här färdighet. Om man dessutom är kritisk till det man skrivit och inte tvekar att formulera om luddigheter blir man med tiden en bättre skribent.

En mycket läsvärd bok om hur man skriver bra texter är *On Writing Well* av William Zinsser (finns som pdf via google). Boken beskriver amerikanska förhållanden och en del av den riktar sig tydligt till den inhemska befolkningen. Övriga delar är däremot helt tidlösa och borde vara ett rättesnöre för oss alla.

I det långa loppet borde också böcker om retorik och inte minst retorikens historia vara matnyttiga.

Rubrik/kapitelnamn

INNEHÅLL

Kapitelnummer

SIDNUMMER

ÖVERSIKTLIGT FÖRST...

...SEDAN DETALJER

ÖVERSIKTLIGT FÖRST...

...SEDAN DETALJER

STRUKTURNIVÅER

Innehåll

FRÅN ÖVERSIKTLIG NIVÅ TILL MER DETALJERAD

1 ALLMÄNT

Vad man kan förväntas sig av texten.

Beskrivningens del I* omfattar en kortfattad presentation av och anvisningar för handhavande av radiostation 10 W Br m/39 och 10 W Br/4 m/39-43. En mera ingående beskrivning av stationens elektriska utförande samt anvisningar för felsökning, trimning och provning, lämpade för speciellt utbildad servicepersonal lämnas i föreliggande del II. 1

* Instruktion för 10 watts bärbar radiostation /4 m/39-43.
" " " " " m/39.

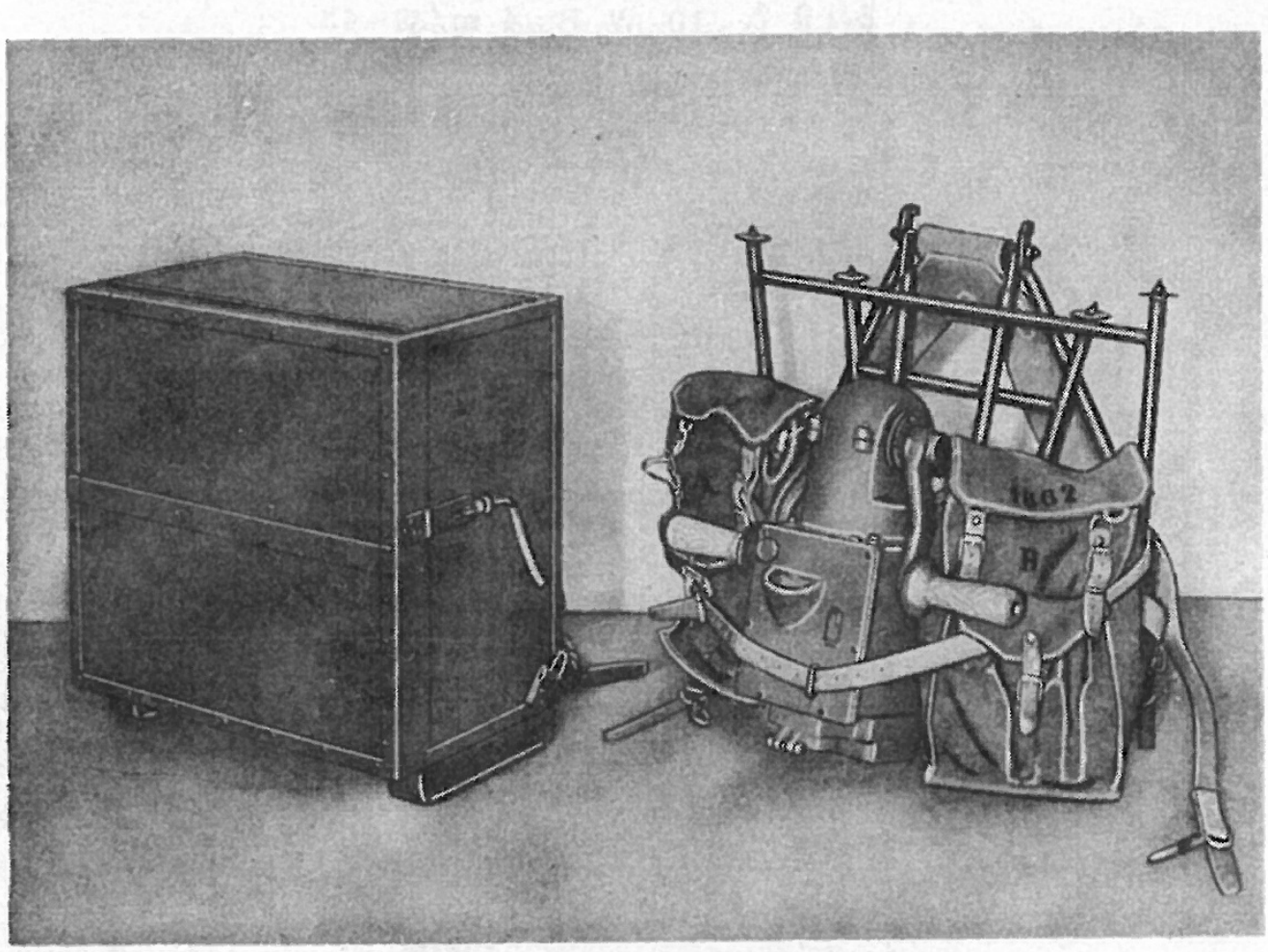


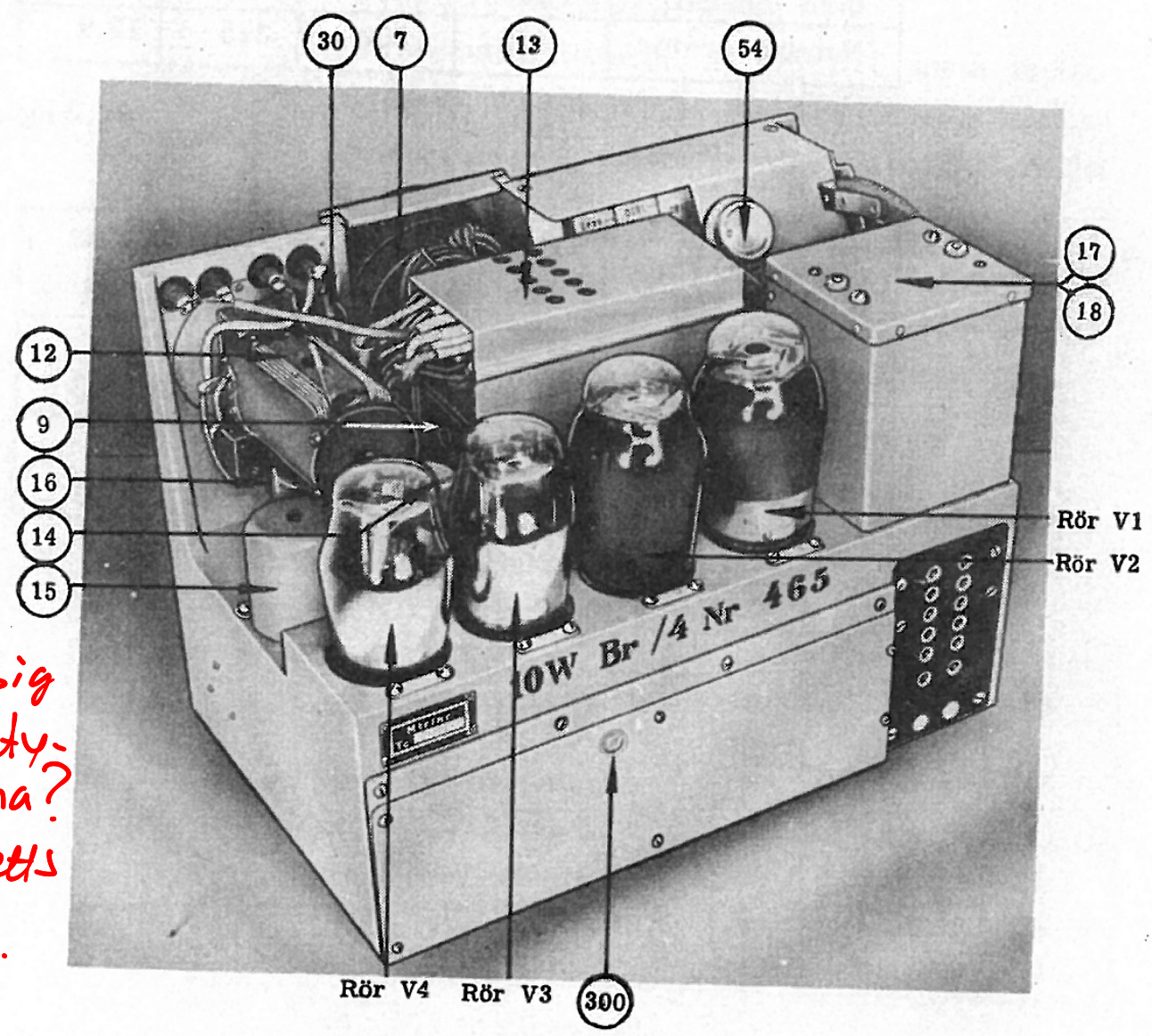
BILD 1 10 W Br m/39

För att förankra läsaren, tydligt ange vad som kommer avhandlas kan en bild/figur vara lämplig.

I allmänhet: Tre rader bildtext — åtminstone!

3 KONSTRUKTION

21 Beträffande stationernas konstruktion och materielens handhavande hänvisas till del I av resp "Instruktion för 10 W Br m/39" och "Instruktion för 10 W Br/4 m/39-43".



Man frågar sig här: Vad betyder siffrorna? Borde anges i bildtexten.

BILD 4 SÄNDAREN SEDD SNETT BAKIFRÅN

Inte bara SÄNDAREN utan också att den i bilden är sedd SNETT BAKIFRÅN

Beskrivande text i opersonlig form

4 FUNKTION

31 I föreliggande instruktion kommer stationens elektriska funktion att behandlas i två avsnitt.

Först lämnas en koncentrerad beskrivning, vilken ansluter sig till stationens blockschemor. Därpå följer en detaljerad beskrivning av samtliga i stationen ingående kretsar. Detaljnumren hänföra sig till kopplingsscheman bil 5 - 13.

KONCENTRERAD BESKRIVNING

32 Antennen, som utgöres av en kastantenn, är gemensam för sändare och mottagare. Medelst ett relä i sändaren (antennreläet) kan antennen anslutas alternativt till sändaren eller mottagaren. Reläet manövreras genom S-M-omkopplaren.

Radiostation 10 W Br/4 och Mt Br m/36-43 äro dessutom utrustade med en stavantenn, vilken är avsedd att användas t ex vid mottagning under marsch.

SÄNDAREN

33 Sändaren erhåller glödspänning (8 V) och anodspänning (350 V) från handgeneratorn. Vid sändning med vågtyp A1 erhåller S-M-reläet manöverspänning från handgeneratorns lågspänningslindning.

Vid sändning med vågtyp A3 erhåller S-M-reläet manöverspänning från stationens nifeackumulatorer (2 st D10 i serie), vilka även mata mikrofonkretsen.

Sändaren har 4 elektronrör: 1 styrrör, 1 effektrör och 2 st mottaktkopplade modulatorrör.

Notera: Blockschema först...

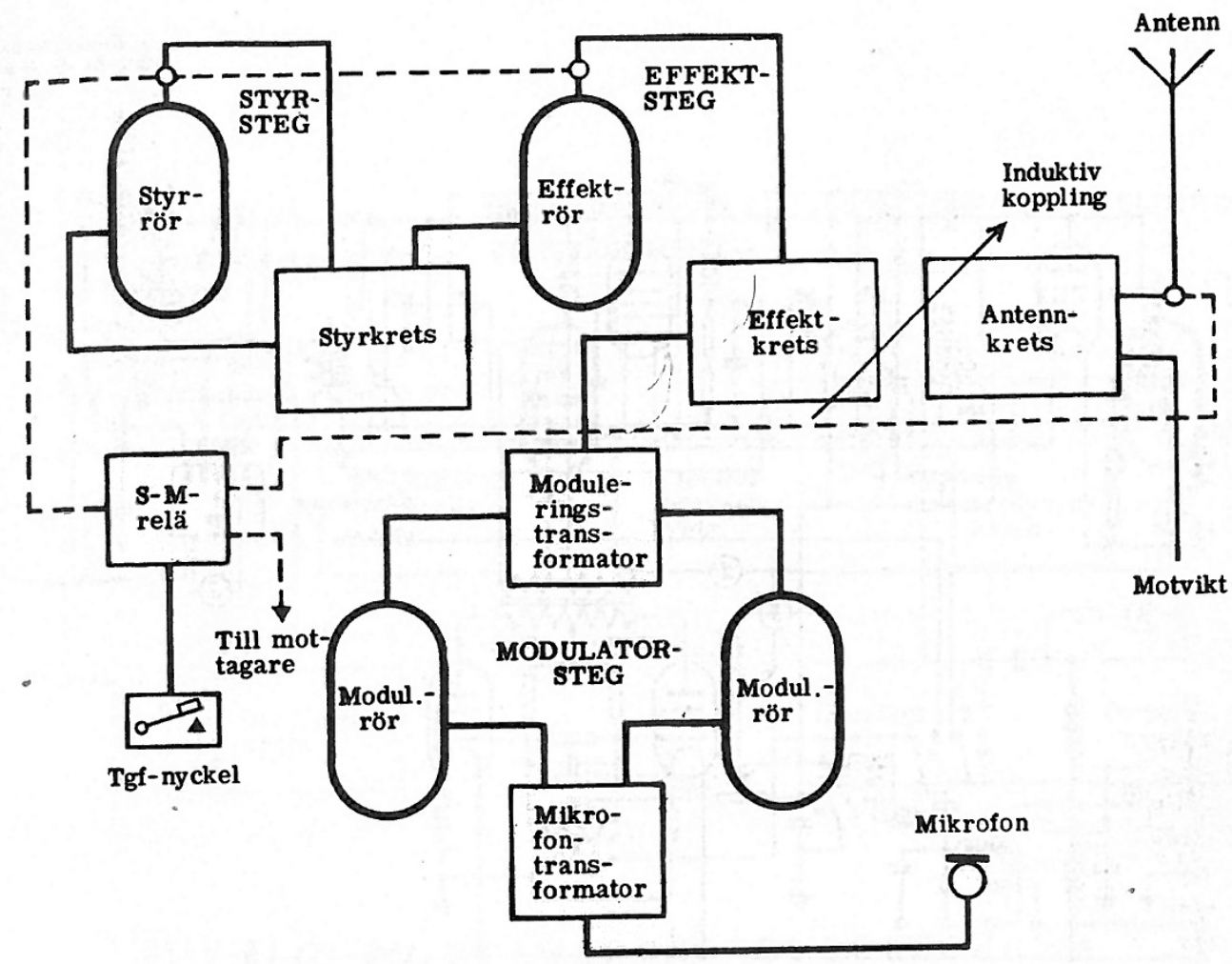


BILD 9 SÄNDARENS BLOCKSCHEMA

Styrsteget, som utgöres av en självstyrd oscillator, arbetar på en frekvens inställbar kontinuerligt mellan frekvensgränserna 2500 och 5000 kp/s. Den genererade spänningen tillföres effektsteget, som är avstämt till samma frekvens som styrsteget. Den i effektsteget erhållna högfrekvenseffekten överföres induktivt via antennkretsen till antennen. 34

Antennen utstrålar energien i form av elektromagnetiska vågor.

Vid sändning med vågtyp A3 överföras talimpulserna från mikrofonkretsen till modulatorsteget, där de förstärkas. De i modulatorsteget förstärkta talfrekventa spänningarna tillföras effektsteget. I effektsteget moduleras den från styrsteget erhållna högfrekvensen varefter denna (den modulerade bärvågen) tillföres antennkretsen enl mom 34. 35

 ... detaljer sedan...

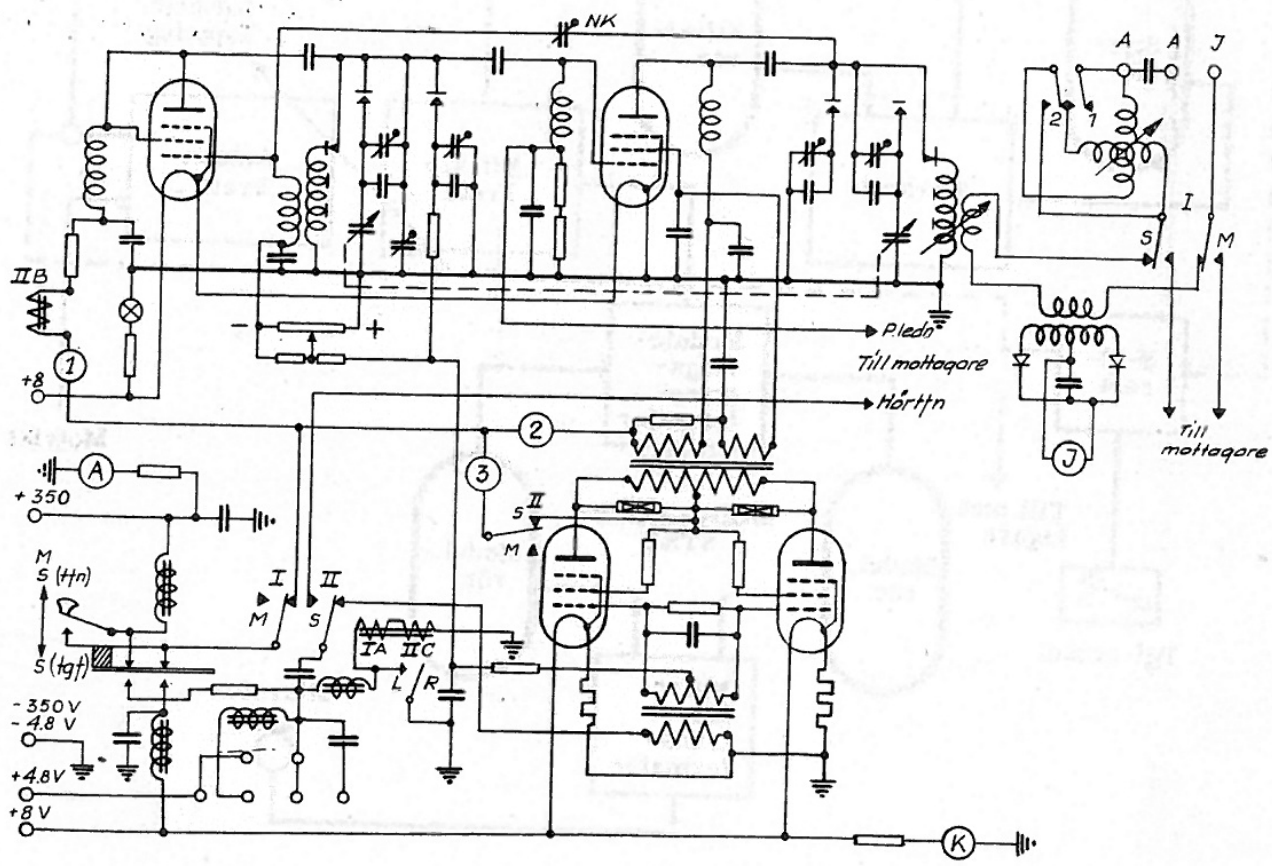


BILD 10 SÄNDARENS PRINCIPSCHEMA

MOTTAGAREN

36 Mottagarrören erhålla glödspänning från stationens nifeackumulatorer (2 st D10 i serie). Anodspänningen erhålles från 1 st 126 V torrbatteri. Till 10 W Br/4 finnes även 1 st 4,5 V ficklampsbatteri, vilket lämnar erforderlig gallerförspänning vid nödmottagning se mom 101.

Mottagaren är en superheterodynmottagare med 4 steg: blandarsteg, MF-steg, detektorsteg och slutsteg.

De signaler, som uppfångats av antennen, kopplas via kabelaget till mottagaren.

Och mottagaren på liknande sätt. Block-schema först...

Funktion

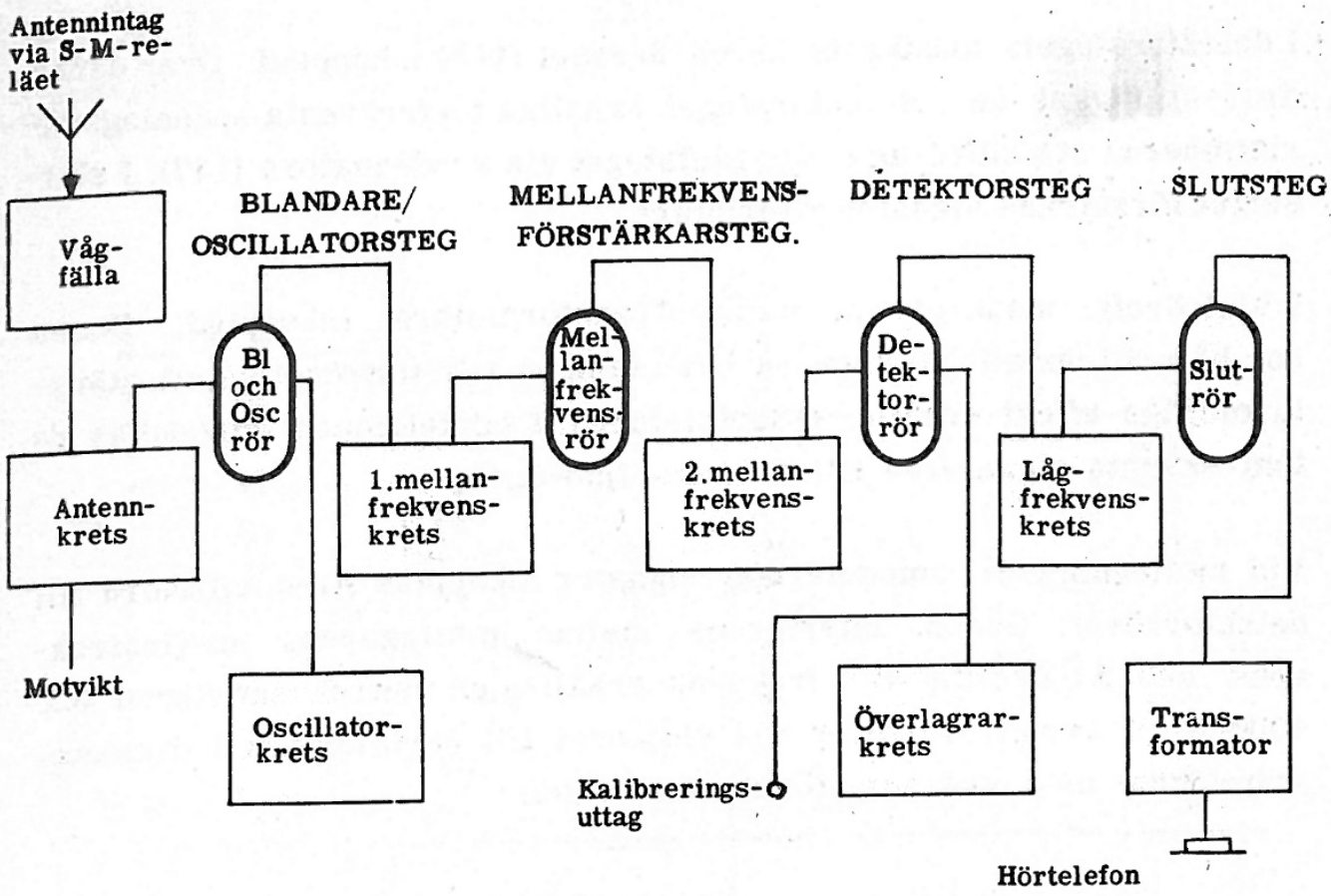


BILD 11 MOTTAGARENS BLOCKSCHEMA

Före mottagarens antennkrets är inkopplad en vågfälla med uppgift att hindra signaler med samma frekvens som mellanfrekvensen (1200 kp/s) att inkomma i mottagaren.

Från antennkretsen överföras signalerna till blandarröret. Den inkommande högfrekvensen blandas här med en annan högfrekvens, erhållen från en lokaloscillator, varvid en skillnadsfrekvens den s k mellanfrekvensen uppstår.

Mellanfrekvensen (MF) överföres från blandarrörets anodkrets till MF-förstärkarrörets styrgaller.

I detektorröret likriktas MF-signalen, varvid en i tonfrekvent takt varierande likspänning erhålles. Efter att ha fullbordat sin uppgift som bärfrekvens för de tonfrekventa signalerna kortslutes högfrekvensen av kondensatorn (detalj 168).

Funktion

I detektorstegets anodkrets är en drossel (115) inkopplad. Över denna drossel uttagas de i detektorsteget erhållna tonfrekventa spänningsvariationerna och tillföras sedan slutsteget via kondensatorn (157). I slutsteget förstärkes signalen ytterligare.

38 I slutrörets anodkrets är utgångstransformatorn inkopplad. Denna har bl a till uppgift att anpassa belastningen till slutröret så att största möjliga effekt erhålles i hörtelefonen. I hörtelefonen omvandlas de tonfrekventa signalerna till hörbara ljudvågor.

39 Vid mottagning av omodulerade signaler inkopplas A1-oscillatorn till detektorröret. Genom interferens mellan mottagarens mellanfrekvens och A1-oscillatorns frekvens erhålles en tonfrekvent signal (ca 1000 p/s). Denna överföres via slutröret till hörtelefonen i överensstämmelse med vad som förut relaterats.

...detaljschema sedan...

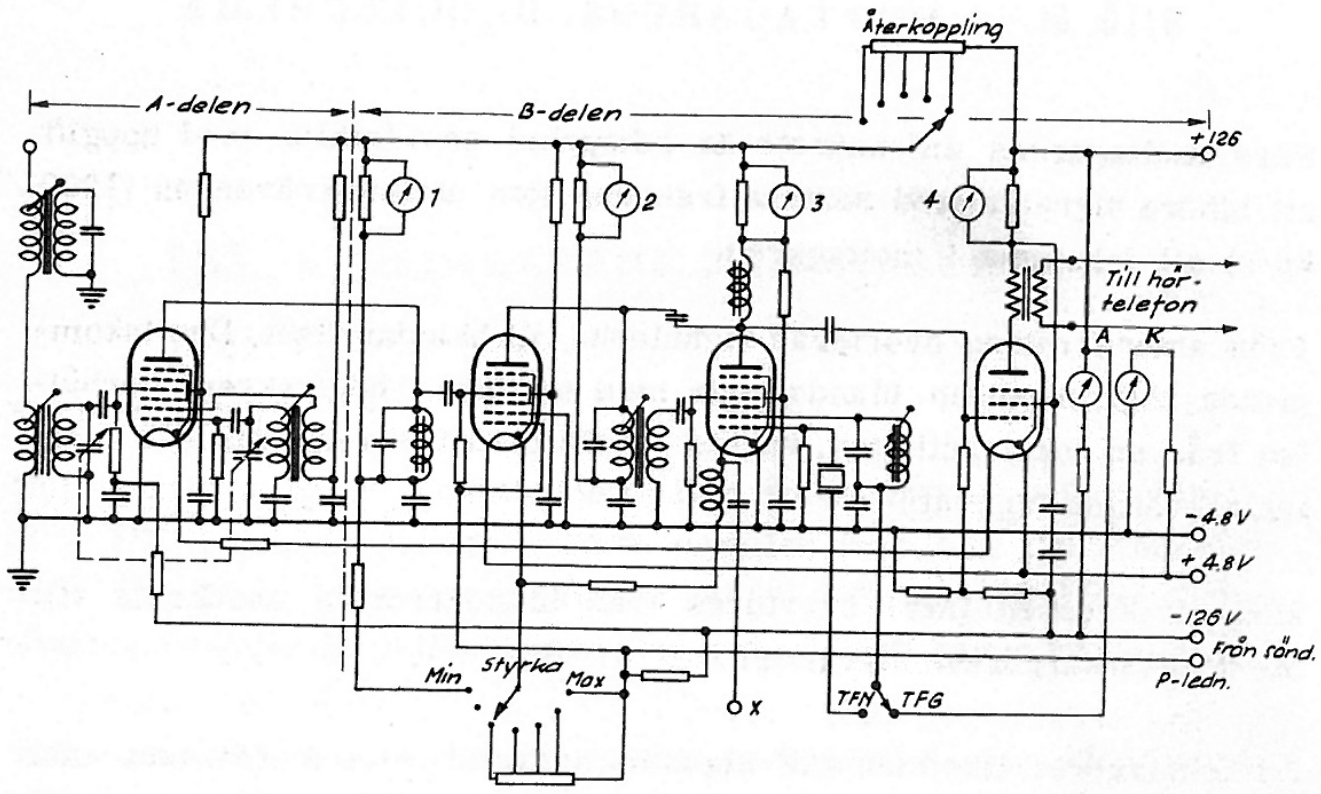


BILD 12 MOTTAGARENS PRINCIPSCHEMA

DETALJERAD BESKRIVNING

SÄNDAREN

Tack vare smygstarten med blockschemana är vi redo att läsa mer och mer detaljerade beskrivningar.

ÄNNU mer detaljer nu!

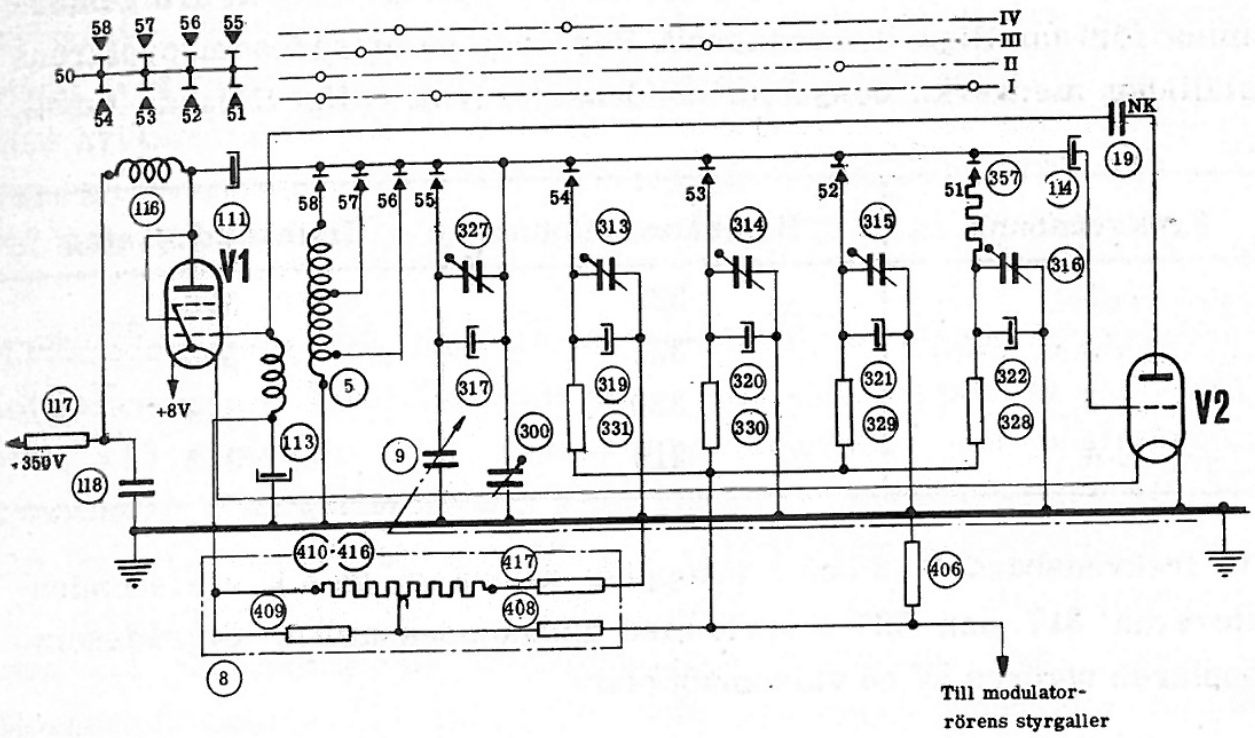


BILD 13 STYRSTEGETS PRINCIPSCHEMA

Styroscillatorn

Styroscillatorn bestämmer sändarens frekvens, som avstämmes med hjälp av kondensatorn 9.

Röret, som har avstämd anodkrets, arbetar med stor negativ gallerförspänning (klass C). Den negativa gallerförspänningen åstadkommes av den ström, som flyter genom potentiometern 410 - 416 och motståndet 417 jämte de parallellt med dessa liggande motstånden 406, 408 och 409. Anodspänningen tillföres röret över motståndet 117 och högfrekvensdrosseln 116.

40

Innehållsförteckning

1. Inledning

Denna rapport utgör den skriftliga dokumentationen av det projekt grupp 9 valt i kursen Mikrodatorprojekt, TSIU51. Gruppen har valt att konstruera en RFID kortläsare med en display.

Blockschema över konstruktionen hittas under Appendix A, och programkoden hittas under Appendix B.

2. Mål

Målet med kursen var att bl.a. lära sig kommunikation mellan två processorer, och att lära sig arbeta i grupp. Vårt mål var att konstruera en RFID kortläsare med en display så att det fungerade som ett passersystem.

Kravspecifikation

- Kommunikation mellan två processorer.
- Läsa in kort med hjälp av en kortläsare (*RFID Detector*).
- Jämföra kortet för att se om det är ett giltigt kort i vår databas.
- Skriva ut korthavarens namn på display (om giltigt).

I mån av tid

- Ha ett klocksystem, med hjälp av en realtidsklocka.
- En logg som sparar vid vilken tid vilket kort användes.
- Lägga till och ta bort kort i databasen.
- Pin-kod.
- Ett avancerat gränssnitt.

3. Kort beskrivning av RFID

RFID är en typ av trådlös dataöverföring mellan en läsare och en s.k. tagg. En tagg är till exempel ett kort med ett litet kretskort som innehåller ett ID-nummer som är unikt för just den taggen. Tekniken använder sig av elektromagnetiska fält och radiovågor från läsaren för att föra över data från taggen.

Det finns två typer av RFID: aktiv RFID, och passiv RFID.
Med aktiv RFID menas att taggen har en egen strömkälla som driver dess inre komponenter, medan passiv RFID får all sin strömförsörjning från läsarens elektromagnetiska fält med hjälp av induktion. Så fort en passiv RFID tagg är nära en läsare kommer den börja sända ut sitt data från sin inbyggda antenn.

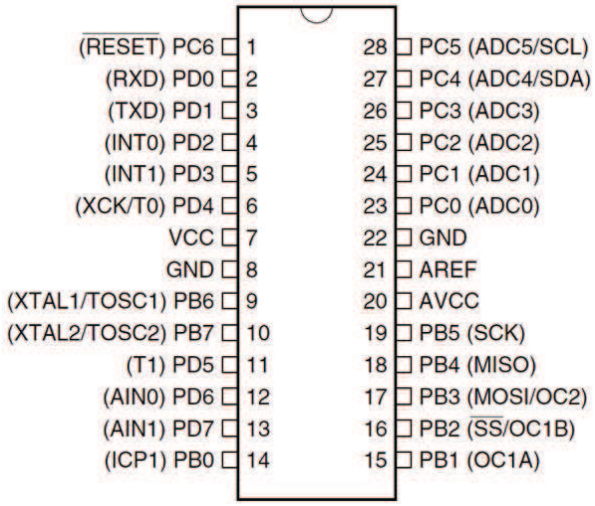
4. Beskrivning av hårdvara

4.1. Beskrivning av ATMega8

ATMega8 är en lågenergi 8-bitars mikrokontroller från ATMEL med en klockhastighet på upp till 8 MHz.

ATMega8 har 8 KB programminne, 512 B *EEPROM* och 1 KB *SRAM*.
Det finns totalt 28 pinnar, varav 23 är programmeringsbara. De kan vi bestämma om de ska skicka eller ta emot data.

EEPROM står för *Electrically Erasable Programmable Read-Only Memory*. Det är helt enkelt minne som inte tappar sitt data när spänningen försvinner.

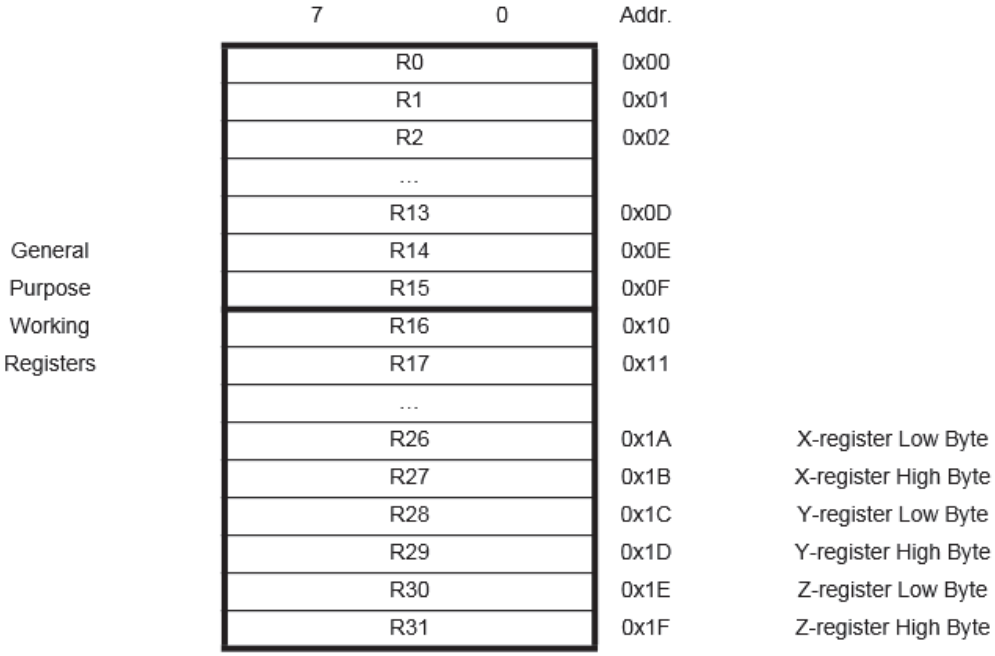


Figur 1 - ATMega8

SRAM står för *Static Random Access Memory*. Det används som ett arbetsminne av strömsnåla processorer.

Det finns 15 register (R16 – R31) tillgängliga. Register R16 – R25 går att namnsätta och är en byte stora.

R26 – R31 består av X-register (R26 – R27), Y-register (R28 – R29) och Z-register (R30 – R31).



Figur 2 - Register

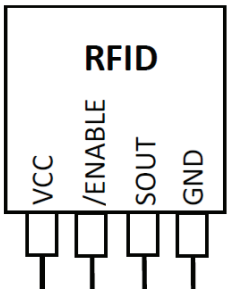
4.1.1. Programmering av ATMega8

Programkoden skrevs i Atmel Studio 6 i assemblerspråk.

Med hjälp av en så kallad ”brännarstation” brändes koden in i processorn. Detta skedde också med hjälp av Atmel Studio 6.

4.2. Beskrivning av Kortläsare

Kortläsaren är tillverkad av Parallax Inc. Kortläsaren har fyra pinnar som måste kopplas för att den ska fungera; *Vcc*, */ENABLE*, *GND* samt *sOUT*.



Figur 3 - Kortläsaren

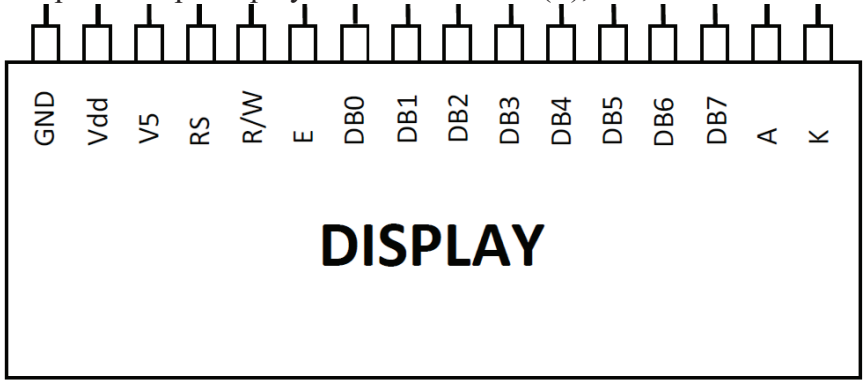
Vcc är strömförsörjningen och ska kopplas till 5 V, och *GND* ska då givetvis kopplas till jord. */ENABLE* bestämmer om läsaren ska vara aktiv eller inte. För att sätta kortläsaren i aktivt läge krävs en låg signal. En hög signal sätter då läsaren i ett inaktivt läge.

När kortläsaren är i aktivt läge och ett kort kommer tillräckligt nära börjar kortläsaren att skicka ut kortets unika ID (10 bytes), start- (0x0A) och slutbyte (0x0D) på ASCII-form genom *sOUT*.

Start- och slutbyte används för att kontrollera att en korrekt data har överförts istället för brus. Datan kommer seriellt med en hastighet på 2400 bits/s, med en startbit (låg) innan varje byte, samt en slutbit som är hög. Kortläsaren kommer att skicka kortets ID om och om igen tills kortet är borttagit från kortläsarens räckvidd.

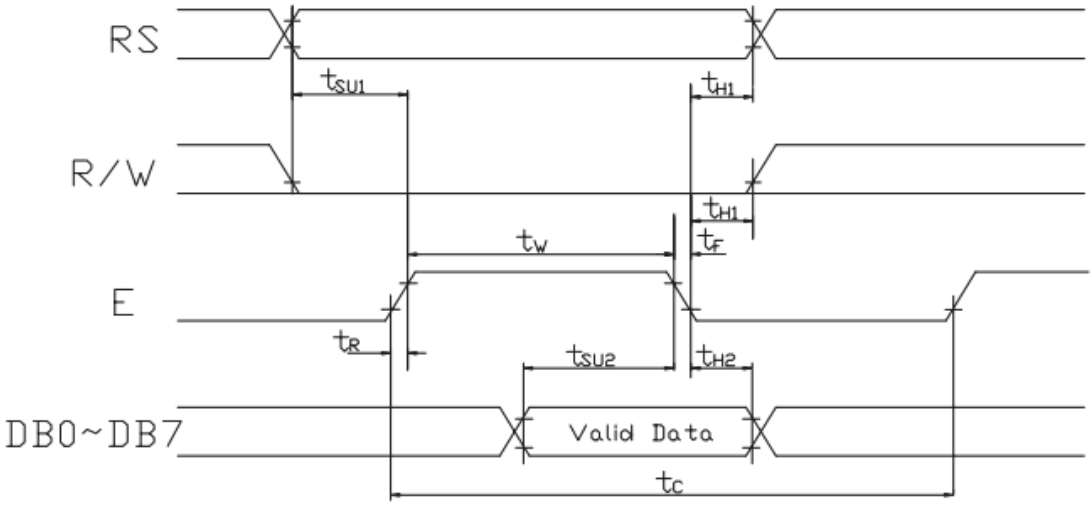
4.3. Beskrivning av Display

Displayen som använts är av typen HD44780. Den har 16 pinnar och två rader à 16 tecken att skriva på. Bland pinnarna finns 8 stycken datapinnar, *DB0 – DB7*. Där läggs data man skickar till eller från displayen. Det finns givetvis *Vdd* samt *GND* också, som då ger ström åt själva displayen, de sista två pinnarna är anod och katod för belysningen för displayen. De tre kanske viktigaste pinnarna på displayen är dock *enable* (*E*), *R/W* samt *RS*.



Figur 4 - Display

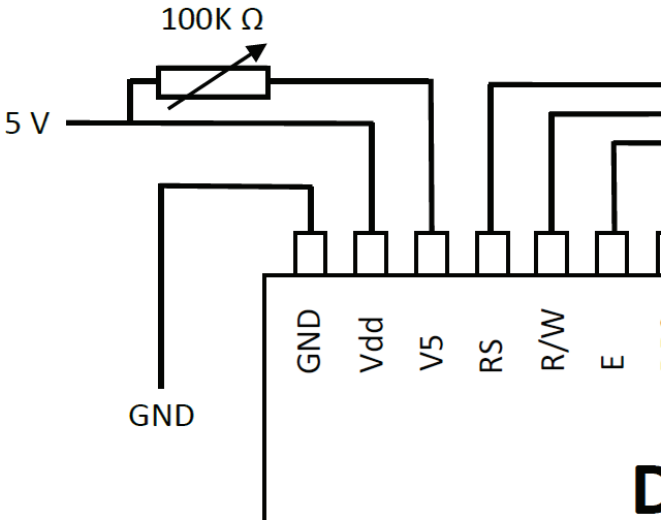
För att skriva till displayen måste *R/W* vara låg, och *enable* måste vara hög. *RS* visar om det är en instruktion eller ett tecken du vill skriva till displayen, där hög betyder data och låg instruktion. När sedan *enable* får låg flank kommer all data att tas upp och behandlas av displayens egna mikrokontroller. Detta beskrivs också med hjälp av följande schema.



Figur 5 - Flanker

DDRAM är displayens egna minne där tecken som skrivits ut sparas. En ”ruta” på displayen motsvarar en byte i minnet. Minnet har även en pekare som pekar på positionen som tecknet skrivs ut på.
Är det ett tecken så kommer detta skrivas ut på den positionen i *DDRAM* som displayens pekare befinner sig på och sedan automatiskt hoppa fram ett steg så att nästa tecken skrivs ut på nästa position.

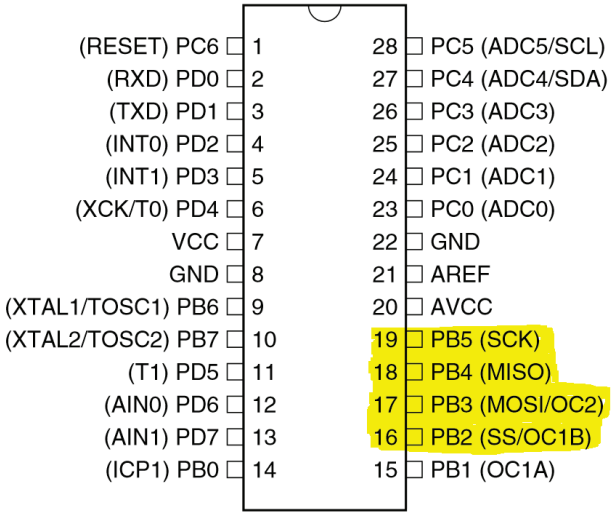
En viktig sak att notera med displayen är också att den behöver en variabel spänning på pinne *V5*, detta är för att kontrasten på displayen ska fungera, d.v.s. svart ska skiljas från “vitt” (i detta fall grönt). Här kopplas förslagsvis en potentiometer på cirka 100 k ohm in på pinnen *V5* som man sedan kan variera så att kontrasten blir bra.



Figur 6 - Potentiometer och display

4.4. Beskrivning av SPI

Enkelt förklarat sker dataöverföring med *SPI* genom att synkront skifta bytes mellan två skiftregister i olika utrustning. All data som är i master processorns skiftregister skiftas över till slave och vice versa. Detta styrs med fyra pinnar: *MISO*, *MOSI*, *SCK* och *SS*.



Figur 7 - SPI på ATMega8

MISO och *MOSI* skickar data mellan slave och master, *SCK* genererar klockpulser och *SS* används för att välja slav och för att nollställa *SPI* logiken.

För att konfigurera och använda *SPI* finns tre register: *SPCR*, *SPSR* och *SPDR*.

SPCR konfigurerar hur den ska uppföra sig (om den ska vara master eller slave), *SPSR* har flagga för avbrott och på *SPDR* läggs/hämtas data som ska överföras/tas emot.

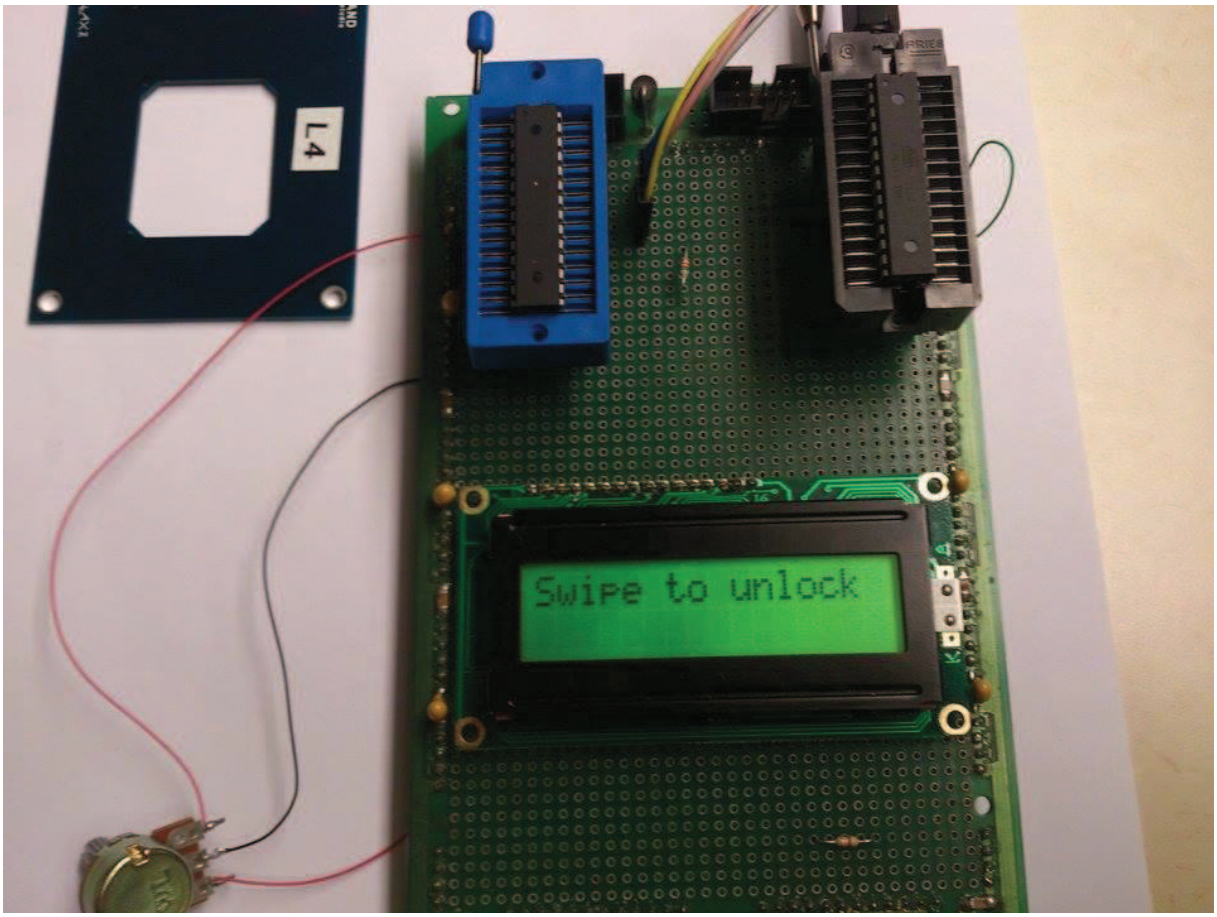
När en överföring är klar sätts *SPIF*-flaggan i *SPSR*. När denna flagga sätts genereras ett avbrott (om *SPI* avbrott och globala avbrott är aktiverade). *SPIF* nollställs antingen genom att avbrottsvektorn anropas, eller att genom först läsa *SPSR* och sedan använda *SPDR*.

5. Konstruktion

När projektet påbörjade hade gruppen två kopplingsdäck att arbeta med. För att utnyttja tiden på bästa sätt delades gruppen upp i två grupper och kopplade var sin krets, t.ex. hade en grupp RFID läsaren och en processor, och den andra gruppen hade displayen och den andra processorn.

Mot slutet av projektet kopplades båda processorerna, displayen och RFID läsaren in på ett och samma kopplingsdäck så att processorerna kunde kommunicera mellan varandra.

Ett kretsschema över hur vi kopplade finns i Appendix A.



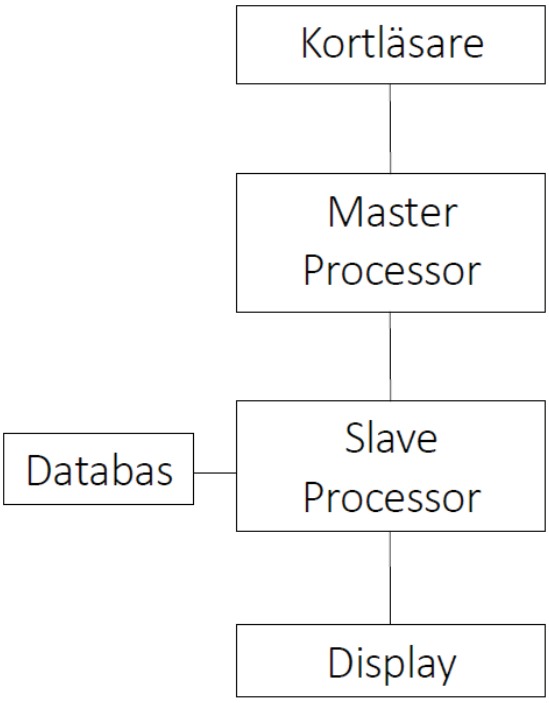
Figur 8 - Den färdiga konstruktionen av vårt passersystem

6. Programvaran

Här förklaras det mer hur master- och slave-processorerna är uppbyggda.

6.1 Master

Master-processorns funktion är att ta emot en ID från RFID-läsaren genom *USART*, kontrollera att det är en korrekt ID genom dess start- och slutbyte och skicka den vidare till slave-processorn med *SPI*. Efter en ID är skickad väntar processorn på att slave-processorn ska arbeta klart.



Figur 9 - Blockschema

6.1.1 USART

Mottagning av data från kort-läsaren sker genom *USART* som är en hårdvarufunktion i ATMega8 för att ta emot och skicka data seriellt. *USART* använder pinnarna *PD0* (*RXD*) och *PD1* (*TXD*). *PD0* används för att ta emot data och *PD1* för att skicka data.

Figur 10 - USART

Kortläsaren och master processorn jobbar inte i samma hastighet. Därför måste man ange vilken hastighet *USART* måste jobba i för att kunna synkronisera med kortläsarens bit-ström. Den här hastigheten anges genom nedanstående formel och sätts i registret *UBRR* som tillhör *USART*.

$$UBRR = \frac{FOSC}{\frac{16}{baud}} - 1$$

F_{OSC} är master-processorns hastighet (8 MHz i vårt fall) och *baud* är kortläsarens hastighet (2400 bits/s). Enligt formeln sätts *UBRR* till 207.
UBRR består utav två 8-bitars register, *UBBRH* och *UBRRL* (de lägsta bitarna i *UBRR*), för att kunna uppnå värden mellan 0 - 4095. Enligt formeln ovan ska *UBRR* sättas till 207, vilket betyder att *UBRRH* ska sättas till noll och *UBRRL* till 207. För att använda pin *PD0* som ingång till *USART* behövs bit *RXEN* (bit 4) ettställas i *UCSRB*. Efter *UBRR* är satt, behöver man ange vilket format bitströmmen kommer i. Detta görs i register *UCSRC*.

Varje byte som skickas från kortläsaren inleds med en låg startbit och avslutas med en hög slutbit. När *UBRR*, *UCSRB* och *UCSRC* är satta kontrolleras värdet på bit *RXC* (bit 7) i *UCSRA*. Om *RXC* är ettställd är *UDR* full. *UDR* är *USART*:s buffer där all inkommen data lagras. *UDR* fylls endast med en byte i taget, vilket gör att man måste hämta datan från *UDR* varje gång *RXC* är ettställd.

6.1.2 Sändning via SPI

Initiering av master görs med hjälp av rutinen `SPI_MasterInit`, vilket även sätter klockhastighet och *SS* pinnen hög (för att tvinga in slave *SPI* i viloläge). `SEND_ID` sköter skickandet av data. Detta sker i följande steg:

1. *SS* sätts låg
2. Byte som önskas skicka läggs på *SPDR*
3. Vänta tills överföring är klar
4. *SS* sätts låg
5. Steg 1 – 4 repeteras tills alla bytes är skickade

Slave-processorn måste hinna bearbeta data som tas emot. Därför har vi små fördröjningar mellan steg 1 och 2, och mellan steg 3 och 4.

6.2. Slave

Slave processorns funktion är att ta emot en ID genom *SPI*, söka databasen efter denna och skriva ut olika meddelande på displayen beroende på om ID hittades. Allt detta sker i en avbrottsrutin.

6.2.1 Mottagning via SPI

Initiering av slave görs med hjälp av `SPI_SlaveInit`. Mottagningen av *SPI* är avbrottsstyrt: när en byte skiftats över anropas avbrottsvektorn PROCESS. Själva mottagningen sker i subrutinen `SPI_read`. Innan mottagningen börjar sätts *PIN5* i *PORTC* låg för att notifiera master om att data bearbetas. Eftersom *SPIF* nollställs när avbrottsvektorn anropas sker mottagningen i följande steg:

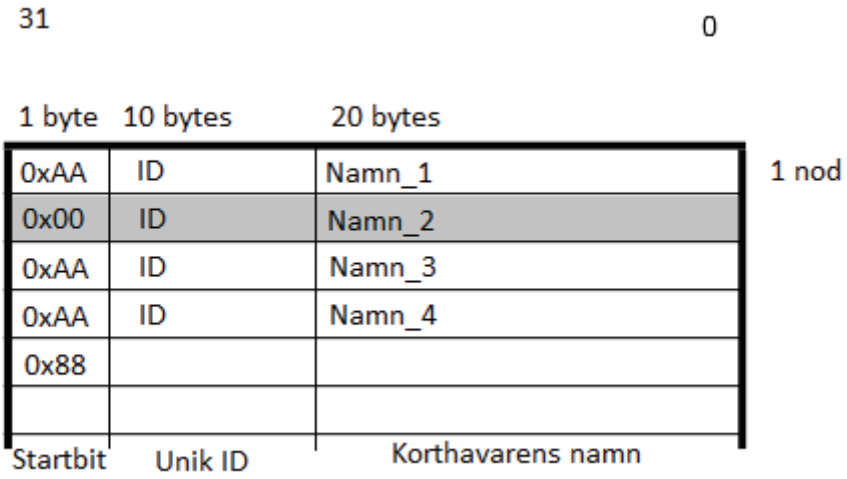
1. Ladda in först byte i *SRAM*
2. Vänta tills nästa överföring är klar
3. Ladda in nästa byte i *SRAM*
4. Steg 2 – 3 repeteras tills alla bytes är mottagna

Efter detta sätts åter igen *PIN5* för att informera master om att en ny ID är redo att tas emot.

6.2.2 Sökning i databasen

Databasens datastruktur är uppbyggd på ett sådant sätt att det är möjligt att modifiera existerande kod så att man både kan lägga till och ta bort kort vid behov.

Alla kort sparas i noder på 31 bytes i processorns *EEPROM*. Varje nod består av tre delar: en verifieringbyte, tio bytes för kortets ID och 20 bytes för kortägarens namn. Verifieringsbyten har värdet 0x88 ifall listan är slut, 0xAA om aktuell nod är aktiv och 0x00 ifall den är inaktiv.



Figur 11 - Databas

Denna struktur underlättar processen att "ta bort" ett kort ur databasen då endast verifieringsbyten behöver ändras till inaktiv. Det resulterar också i att man kan spara nya kort på oanvända noder istället för att expandera listan varje gång ett nytt kort behöver läggas till.

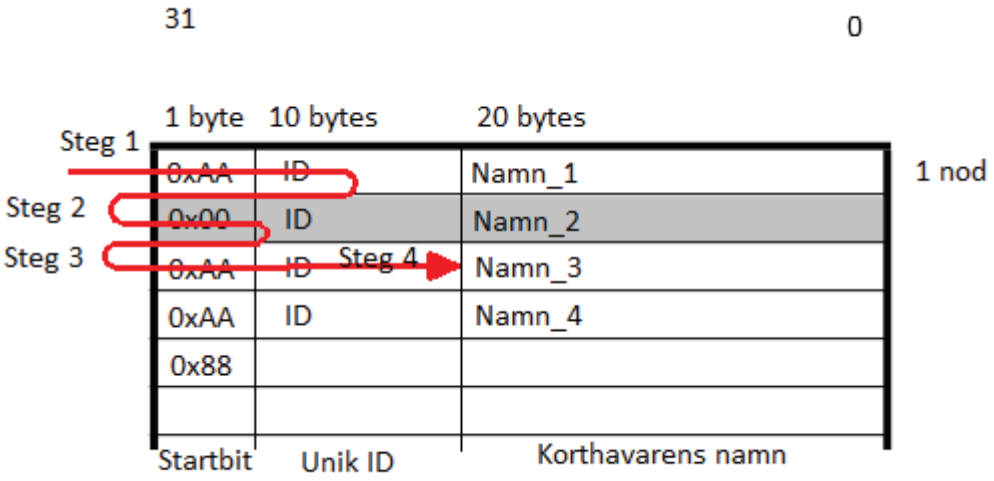
Sökning och hämtning av namn hanteras av `LOOKUP_ID`, vilket använder subrutiner som inkluderas i `List.asm`. Dessa subrutiner gör det möjligt att arbeta med datastrukturen, och innefattar följande funktioner:

- Undersöka om listan är tom. [`LIST_EMPTY`]
- Hoppa till nästa nod om möjligt. [`NEXT_NODE`]
- Jämföra ID i *SRAM* med ID i nod. [`COMPARE_ID`]
- Ladda namn från nod till *SRAM*. [`LOAD_NAME`]

Med hjälp dessa kan `LOOKUP_ID` söka igenom listan, meddela huvudrutinen om id hittades, och ladda eventuellt namn till ett datasegment.

Ett exempel på hur det ser ut med hjälp av föregående bild skulle det se ut såhär om vi söker efter Namn_3 i dessa följande steg:

1. Kolla efter startbit, sedan jämföra ID. ID stämmer inte; hoppa till nästa nod.
2. Kolla efter startbit. Nod är inaktiv; hoppa till nästa.
3. Kolla efter startbit. Jämför ID. ID stämmer!
4. Ladda namn från nod till *SRAM*. Avsluta.



Figur 12 - Sökning i databasen

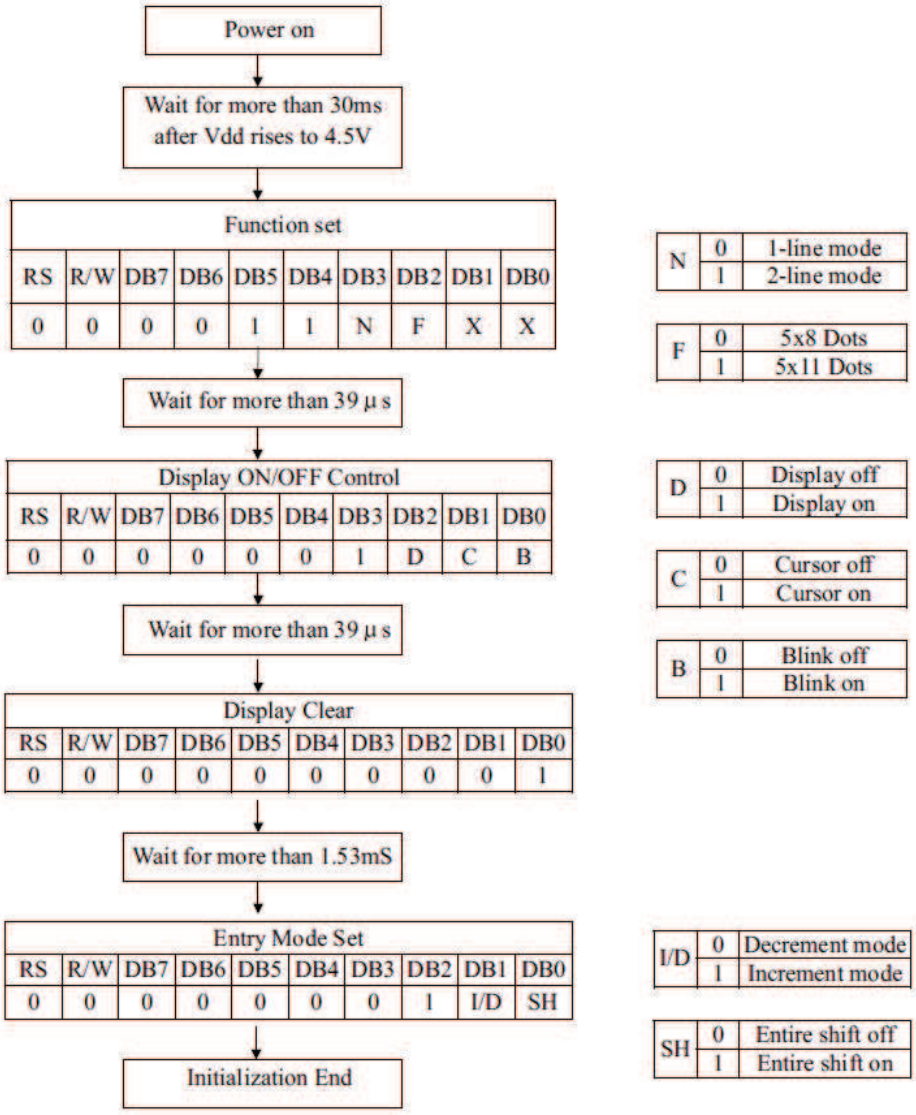
Mer detaljerad information återfinns i bilagor i form av kod.

6.2.3 LCD

För att initiera displayen krävs det att det skickas sju stycken instruktioner med rätt fördröjning emellan dem. Detta är initieringssekvensen som måste göras:

1. Vänta 30 ms efter att spänningen nått 4,5 V
2. Skicka 0x30 tre gånger.
3. Välja vilken funktion displayen ska ha. Man kan välja mellan en eller två rader och vilken upplösning tecknen ska ha. Vår display behöver visa två rader och 5x8 punkter.
4. Välja om displayen ska vara på/av, pekare på/av, pekare blinkning på/av.
5. Rensa displayen
6. Välja hur skrivning ska gå till. Om den ska inkrementera eller inte och om "*Entire Shift*" ska vara på eller av.

Detta illustreras på nästa sida med hjälp av en bild.



Figur 13 - Initieringssekvensen för displayen

När vi testade displayen hade den en tendens att inte laddas upp korrekt. Detta löstes genom att öka fördröjningen i steg två.

Mellan varje instruktion måste det vara en viss fördröjning, 40 µs räcker oftast.

Displayens drivrutiner är uppbyggda på ett sådant sätt att man enkelt ska kunna skriva till den. De två kanske viktigaste rutinerna som är en del av displayens drivrutiner är `LCD_ascii` och `LCD_cmd`. Beroende på vilken av dem som du kallar på så kommer du antingen skriva ut ett ascii-tecken på displayen, eller ge den ett kommando, till exempel som att rensa displayen.

De är väldigt enkelt uppbyggda och det enda som skiljer dem åt är en kodrad. `LCD_ascii` sätter *RS*-flaggan hög och kallar på `LCD_write`, till skillnad från `LCD_cmd` som sätter *RS*-flaggan låg och sedan kallar på `LCD_write`. `LCD_write` tar datan som lagts i registret "data" och lägger ut det på *portD* (displayens datapinnar). Denna metod gör det enkelt att skilja på när man skriver ett kommando och när man skriver data till displayen.

För att underlätta programmerande och göra koden mer lättläst har vi två stycken standardmeddelanden; `LCD_welcome` samt `LCD_accessdenied`. Dessa har vi hårdkodat in att rensa displayen och sedan skriva "*Welcome*" respektive "*ACCESS DENIED*" på första raden.