Institutionen för Informationsteknologi
Digitala projekt, lp3 2004
2004-03-02

Nummerassistenten och presentatören

Smeagol

Grupp 8
Philip Theorin
Oscar Ekbladh

Abstract

The purpose of the course Digital Systems, Project Laboratory is to design a digital prototype. Our choice was to construct the caller ID Smeagol. First we defined requirements for our prototype and except the traditional caller ID features, Smeagol has the feature to be able to play specific sampled sounds depending on who is calling. The Motorola 68008 CPU was chosen to be the heart of Smeagol and the designing of the hardware begun. After a few weeks we had enough working hardware to be able to start writing the software. Today Smeagol has been running smoothly, without any problems for about 14 hours and all our initial requirements are fulfilled.

Innehållsförteckning

1. Inledning

Kursen digitala projekt går det ut på att under sju veckor designa och bygga en prototyp för vidareutveckling samt att göra en dokumentation av arbetet. Valet föll på att designa en nummerpresentatör vid namn Smeagol, vars utmärkande drag är att den spelar en låt kopplat till det nummer som ringer.

Rapporten inleds med en specifikation av den färdiga prototypen (kap 2) och följs av en beskrivning av hur arbetet fortskridit (kap3). I kapitel 4, 5 och 6 beskrivs sedan hårdvara, adresskodning och mjukvara mer ingående. Efter dessa följer resultat och sammanfattning samt en referenslista. Som bilagor finns kopplingsschema och programkod eller hänvisning till Internetadress med motsvarande material.

2. Specifikation

Nummerpresentatör som spelar upp olika ljud beroende på vem som ringer.

Möjlighet att lagra 2016 olika telefonnummer i en telefonbok och att koppla dessa oberoende till olika ljud och digital utsignal

Möjlighet att programmera in egna telefonnummer med hjälp av dator i ett användarvänligt grafiskt gränssnitt.

Display som visar de 254 senast mottagna nummerna och tidpunkten de kom in.

Möjlighet att lägga in 7 egna ringsignaler.

Klocka som visar datum och tid.

Fem användarvänliga tryckknappar.

Jukebox – Spela dina egna favoritlåtar.

Fyra digitala utgångar – koppla in dina egna favoritprylar till Smeagol®, det kan vara bilen, båten, husvagnen - vad du själv vill göra det till, möjligheterna är obegränsade.

3. Utvecklingsarbetet

Vecka 1

Vi började planering av vad vi skulle göra och kom fram till att vårt ursprungliga projekt var för komplicerat.

Vecka 2

Vi bestämde oss för att göra en nummerpressentatör som spelar ljud och ritade upp ett preliminärt kopplingsschema i PowerLogic.

Vecka 3

Vi kompletterade schemat med latch, display och klocka och fick ut verktygslådan efter att ha visat upp det. Byggandet kunde börja. Anslutningar till matning och jord samt för att programmera Latticen ordnades. Ett första Latticeprogram för att styra CS, Output Enable/Write Enable till RAM-minnet skrevs och testades.

Vecka 4

Vi fick ut vårt IT-68 utvecklingspaket och testade RAM-minnet. Det fungerade perfekt. Vi kopplade in displayen men såg ingen text. Det visade sig bero på att V_0 som styr skärpan var felinställd och att någon kontakt till displayen inte var så bra som väntat. Vi programmerade flash-minnet, kopplade in det och fick det att fungera. Vi har också gjort en flash-editor för att kunna lägga in de data vi vill ha i flash-minnet. Vi kopplade upp realtidsklockan och eftersom vi var tvungna att fördröja kretsens DTACK så var vi tvungna att bygga klocka till processorn också. Två tryckknappar sattes fast och kopplades till latchen. D/A-omvandlaren kopplades in men det visade sig att det var svårare än väntat att koppla den till högtalaren. Efter en stunds kopplande gav vi upp och kopplade in pc-högtalarna och ljuv musik spred sig i salen.

Vecka 5

Nu fungerar det mesta av hårdvaran och vi fokuserar oss på att skriva program. Menysystemet implementerades och testades. Programkod skrivs för att styra de olika kretsarna. Vi försöker lösa problemet med att vi inte får ut något ljud ur vår egen högtalare. Vi försöker lösa problemet med fler OP men den i databladet föreslagna kopplingen med summator efter D/A-omvandlaren fungerar inte och det visar sig att OP-förstärkaren bara kan leverera 20 mA på utgången, vi behöver 20 gånger mer. Lösningen blir att omvandla utsignalen från D/A-omvandlaren till en spänning och att med hjälp av en transistor på förstärkarutgången driva högtalaren. Latch till den digitala utgången kopplades in.

Vecka 6

Nu är det dags att koppla in DTMF-mottagaren och för att det ska fungera så aktiverar vi avbrott. Fungerar bra med bara avbrott aktiverat för DTMF-mottagaren men inte när vi kopplar på avbrotten för realtidsklockan. När vi kom på att vi var tvungna att acka realtidsklockan så fungerade detta också. Vi hade vissa konstiga problem också så efter lång tids sökande visade sig vara problem med minnet. Vi sätter fast det sista på kopplingsbrädet och allting att fungerar faktiskt rätt bra när vi knyter samman programdelarna också.

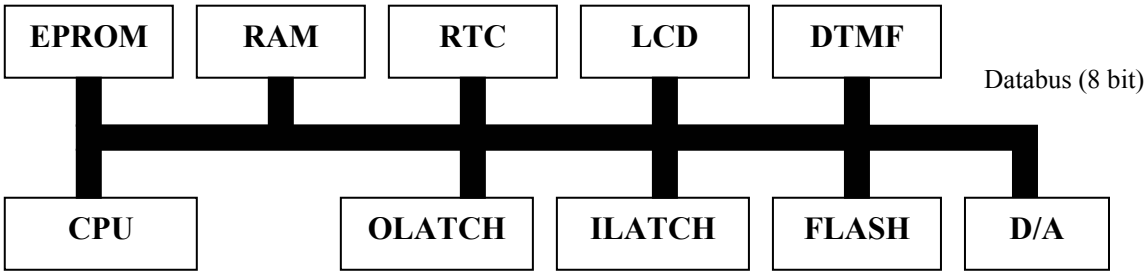
Vecka 7

Rapporten skrivs och koden snyggas till. Nummerpresentatören fungerar stabilt förutom att lite konstiga saker ritas upp på displayen ibland.

4. Hårdvara

Alla komponenter löddes eller virades fast på en kopplingsplatta. För att undvika lång jordväg har extra kablar dragits för att efterlikna ett jordplan. Avstörningskondensatorer har kopplats in på lämpliga ställen nära matningen till de olika kretsarna. Ett komplett kopplingsschema finns att beskåda i Appendix B.

Komponenterna använder en gemensam 8-bits databuss som figur 4.1 visar. Programmerbar logik (lattice) användes för adressavkodning.



Figur 4.1. Databuss

4.1 CPU

Valet föll på Motorolas mikroprocessor MC68008 som är en 32 bitars processor med en 8 bitars databuss och möjlighet att adressera 1 MB.

4.2 EPROM

I den här typen av minne lagras programmet. Valet föll på 27C64 som är ett EPROM på 8 kB och det är tillräckligt för att lagra all kod som behövs för nummerpresentatören.

4.3 RAM

I den här typen av minne lagras variabler och stacken, eftersom det inte handlar om några större mängder data så föll valet på HM6462BI som är 8 kB stort.

4.4 D/A-omvandlare

En 8 bitars D/A-omvandlare vid namn TLC7524 har använts. Den har ström som utsignal och med hjälp av en resistor omvandlas signalen till en spänning innan förstärkarsteget och högtalaren.

4.5 Realtidsklocka

För att hålla reda på tiden används en realtidsklocka, MM58274. Realtidsklockan används också till att generera avbrott varje tiondels sekund för att läsa av knapparnas status.

4.6 Display

En alfanumerisk lcd-display, LM162XXX, med 16*2 teckens storlek används för kommunikation ut mot användaren. Displayen använder sig av en teckenuppsättning som liknar ASCII.

4.7 DTMF Mottagare

Det behövdes en DTMF mottagarkrets för att avkoda vilka nummer det är som ringer, valet föll på MT7870D.

4.8 Tryckknappar

För kommunikation från användare till nummerpresentatören används 4 stycken tryckknappar plus en resetknapp. Tryckknapparna är kopplade till databussen via en latch, 74HC573N.

4.9 Digital utgång

Digital utgång för kommunikation med omvärlden finns i form av en latch 74HC573N.

4.10 Flash-minne

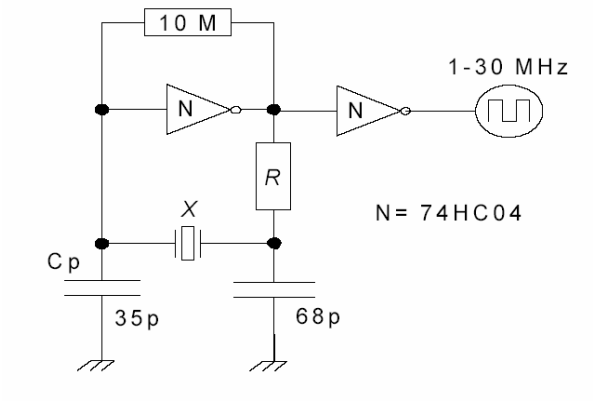
Ett flash-minne används för att lagra telefonnummer och ljudet som ska spelas upp när någon ringer. Flashen är en AM29F0408 på 512 kB och det ger en möjlighet att spara ungefär en minuts musik

4.11 Klocka

Klockan byggdes enligt nedanstående koppling (figur 4.2). Frekvensen valdes till 8 MHz och R valdes enligt

$$R = \frac{10^4}{f} - 300$$

där f anges i MHz.



Figur 4.2. Klockkrets

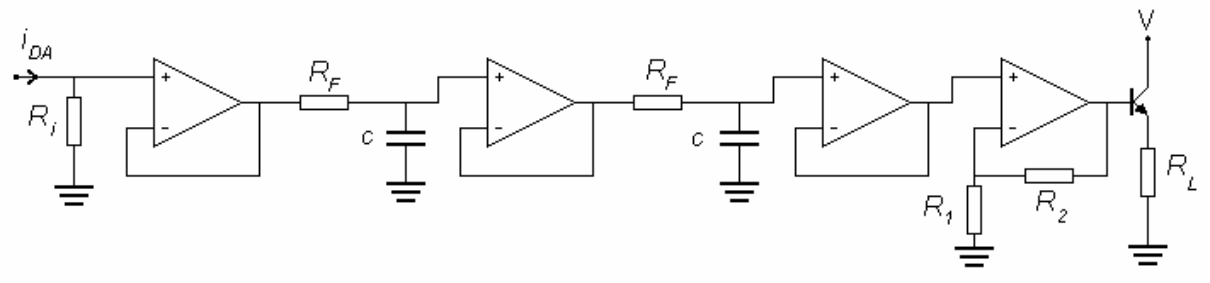
4.12 Förstärkarkrets

Förstärkarsteget används för att filtrera bort övertoner från D/A-omvandlaren och till att driva högtalaren. Förstärkaren är kopplad enligt figur 4.3 och består av LM324 som har fyra inbyggda operationsförstärkare och av en transistor, BD 175, samt resistorer och kondensatorer till filter och återkoppling. I det första steget omvandlas utsignalen från D/A-omvandlaren från en ström till en spänning med hjälp av R_i och därefter är det en spänningsföljare. Efter det kommer det två likadana filter följt av en spänningsföljare. Eftersom ljudet är samplat med frekvensen 8 kHz så har filterna brytfrekvensen 4 kHz och värdena på R_F och C har valts enligt:

$$f = \frac{1}{2\pi RC}$$

Sista steget är en spänningsförstärkare med en transistor på utgången för att orka driva högtalaren (R_L). Förstärkningen bestäms av

$$A = 1 + \frac{R_2}{R_1}$$



Figur 4.3. Förstärkarkrets

4.13 Logik

När det ska göras om och göras rätt är det bra med programmerbar logik. Lattice 1016E med 32 in- och utgångar var lagom stor för nummerpresentatören. Logiken används för adressavkodning och för att styra realtidklockans och DTMF-mottagarens avbrott till processorn.

5 Adresskodning

5.1 Adresskodning av komponenter

De fem mest signifikanta adressbitarna A19-A15 avkodas i Lattice-kretsen för att styra vilken krets som ska använda databusen. Tabell 5.1 visar på vilka adressområden kretsarna är aktiva. Strukturen tillåter utökning av både RAM och ROM upp till 64kB.

Tabell 5.1. Adresskodning, åtkomst (R/W) samt avbrottsgenerering (I)

Startadress	Slutadress	R/W, I	Beskrivning
0x00000	0x01FFF	R	8kB ROM
0x10000	0x11FFF	R/W	8kB RAM
0x20000	-	W	D/A
0x28000	0x28010	R/W I	RTC (realtidsklocka)
0x30000	0x30001	R/W	Display
0x38000	-	R I	DTMF
0x40000	-	R	Latch (input, knappar)
0x48000	-	W	Latch (output, digitala utgångar)
0x80000	0xFFFFF	R	Flash

5.2 Flashstruktur

Flash-minnet kan lagra 512kB och består av åtta 64kB-block som kan raderas separat. I de ursprungliga planerna fanns idéer om att använda en flexibel allokeringstabell för ljudsampler och enbart avvara 8kB av minnet för att spara nummer på. Dessa idéer övergavs dock på grund av flashens egenskaper och för att göra strukturen enklare.

5.2.1 Övergripande struktur

Flash-minnets första 64kB-block används för att associera nummer till namn, ljud och utsignal. De övriga sju blocken används för att lagra ljudsampler.

Tabell 5.2. Övergripande flashstruktur

Startadress	Beskrivning
0x00000	Nummer och inställningar
0x10000	Sampleblock 1
0x20000	Sampleblock 2
0x30000	Sampleblock 3
0x40000	Sampleblock 4
0x50000	Sampleblock 5
0x60000	Sampleblock 6
0x70000	Sampleblock 7

5.2.2 Nummer och inställningar

De allra första 32 byten i Nummer och inställningar-blocket är definierade enligt tabell 5.3.

Tabell 5.3. De första 32 byten i flashminnet (Header)

Adress	Antal byte	Beskrivning
0x00000	0x02	Antal NumberChunks (definierade nedan)
0x00002	0x1E	Padding, används i editorn för att avgöra om en sparad binärfil är en giltig fil eller ej.

Efter detta 32-byteblock följer alla NumberChunks som består av 32 byte och associerar ett visst nummer med ett namn, ljudspår och värde på den digitala utgången. Hur ett NumberChunk är definierat framgår i tabell 5.4.

Tabell 5.4. Definition av ett NumberChunk

Offset	Antal byte	Beskrivning
0x00	0x10	Nummer (ASCII-kodat)
0x10	0x0E	Namn (ASCII-kodat)
0x1E	0x01	Inställningar där bitarna är definierade enligt: xxxxrsss Där xxxx representerar värdet på den digitala utgången, r är reserverad och sss representerar vilket ljud som ska spelas upp. Då sss = 000 spelas inget ljud alls. Övriga sampler är kodade binärt där första sampeln har värdet 001 och sista har värdet 111.
0x1F	0x01	Reserverad för senare bruk.

Förutom helt användardefinierade NumberChunks finns ett antal speciella NumberChunks som ska definiera inställningar för följande typer av nummer:

- Dolda nummer
- Okända nummer

Sammanfattningsvis beskrivs var de olika delarna är placerade i Nummer och inställningar-blocket i tabell 5.5. Med denna struktur finns det plats för 2016 användardefinierade nummer.

Tabell 5.5.

Adress	Antal byte	Beskrivning
0x00000	0x20	Header
0x00020	0x20	SpecialNC: Skyddat nummer
0x00040	0x20	SpecialNC: Okänt nummer
0x00200	0x20	Första användardefinierade NumberChunk

5.2.3 Ljud

Flash-minnets övriga sju 64kB-block används för att lagra ljudsampler. De första 16 byten i varje block består av en ASCII-kodad titel och resterande dataarea består av ljudsampler vilket framgår i tabell 5.6.

Tabell 5.6

Adress	Antal byte	Beskrivning
0x0000	0x0010	Ljudtitel (ASCII-kodad)
0x0010	0xFFE0	Alla ljudsampler, där varje sampel är 8 bitar lång och saknar tecken, vilket innebär att 0x00 står för lägsta signalnivån och 0xFF står för den högsta. Sampler från standard 8kHz, 8 bitars PCM-kodade wave-filer går att använda.

6 Mjukvara

En hel del programkod skrevs till detta projekt. Till hårdvaran skrevs programkod i assembler och i C. Dessutom skrevs en editor till flashminnet i Visual Basic 6. Innan programmeringen påbörjades definierades flashstrukturen (se avsnitt 5.2) och vilka tillstånd menysystemet skulle ha. Programmen är nedbrutna i små, generella funktioner för enkel återanvändning.

6.1 Assemblerkod

Huvudprogrammet är i huvudsak skrivet i C, men en del är skrivet i assembler. Dessa funktioner är listade i tabell 6.1. Förutom pmain som ligger på adress 0 och därmed kommer att köras vid en uppstart av processorn, är funktioner som hanterar avbrott skrivna i assembler. Fullständig kod finns att läsa i Appendix E.

Tabell 6.1. Kort beskrivning av assemblerfilerna

Filnamn	**Beskrivning**
pmain.68k	Initierar stackpekaren, globala datapekaren och avbrottsvektorer. Den nollställer sedan alla dataregister och gör ett subrutinanrop till _main.
exp2.68k	Avbrottshanterare för avbrottsnivå 2. Vid ett avbrott sparas alla register undan på stacken, varefter c-funktionen _exp2 anropas. Därefter återställs alla register.
exp5.68k	Avbrottshanterare för avbrottsnivå 5. Vid ett avbrott sparas alla register undan på stacken, varefter c-funktionen _exp5 anropas. Därefter återställs alla register.
intdis.68k	Avaktiverar alla avbrott.
inten.68k	Aktiverar alla avbrott.
inthalfdis.68k	Tillåter avbrott från avbrottsnivå 5 och uppåt.

6.2 Huvudprogram

Huvudprogrammet är skrivet i ANSI C och finns i fullständig form i Appendix C. I kompilerad form tar programmet i skrivande stund 5.6kB, vilket innebär att det finns knappt 1.5kB kvar i ROM-minnet för vidareutveckling.

6.2.1 Funktioner

I tabell 6.2 är samtliga funktioner i huvudprogrammet, lotr.c, beskrivna.

Tabell 6.2. Funktioner

Funktionnamn	**Beskrivning**
Realtidsklocka	
void init_rtc();	Initierar realtidsklockan
unsigned char read_rtc(int reg);	Läser från ett register I RTC:n
void write_rtc(int reg, unsigned char data);	Skriver en byte till angivet register i RTC:n
void write_date();	Skriver alla värden i datum/tid vektorn till RTC:n.
void read_date();	Läser datum och tid från RTC:n
int rtc_data_changed();	Returnerar 0 om inget ändrats sedan senaste läsning, annars ett värde skilt från noll.
void rtc_int_dis();	Aktiverar generering av avbrott från RTC:n.
void rtc_int_en();	Avaktiverar avbrottsgenerering.
Display/Print-funktioner	
void clr_lcd();	Rensar displayen.
void init_lcd();	Initierar displayen.
void wait_lcd();	Väntar på att föregående instruktion blivit utförd.
void hide_cursor();	Stänger av markör.
void show_cursor(short int row, short int pos);	Visar markör på specificerad rad och tecken.
void println0(char s[]);	Skriver en sträng till övre raden.
void println1(char s[]);	Skriver en sträng till undre raden.
void print(unsigned char c, unsigned short int row, unsigned short int pos);	Skriver et tecken till en specific position på displayen.
void print_from_mem(unsigned long addr, unsigned short int row, unsigned short int pos, int nbrChars);	Läser *nbrChars* antal tecken från adress *addr* och skriver till rad *row* med första tecknet på teckenplats nummer *pos.*
void printbyte(unsigned short int b, unsigned short int row);	Skriver ut bitmönster för en variabel. Används för tillfället inte längre, men var smidig vid debugging.
void printhex(unsigned long int l, unsigned short int row);	Skriver ut en variabels värde i hexadecimal notation.
void print_date(unsigned char d[], unsigned short int row);	Skriver ut datum och tid från en vektor.
Latchar	
unsigned char get_latch();	Läser av vilka knappar som är nedtryckta.
void set_latch(unsigned char c);	Sätter värdet på de digitala utgångarna.

Ljud	
void play_sound(int songNbr);	Spelar upp ett ljud, där *songNbr* är ett värde mellan 1 och 7.
Kontakthantering	
unsigned int nbr_contacts();	Returnerar hur många användardefinierade nummer som finns lagrade i flashen.
unsigned long find_contact(char number[]);	Söker efter ett nummer i flashen och returnerar adressen till det NumberChunk det är associerat med. Finns inte numret definierat i flashen returneras 0x40, vilket innebär okänt nummer.
Loggade nummer (för mer ingående information se avsnitt 6.2.3)	
void init_log();	Initierar log-listan
unsigned char last_log();	Returnerar index till senaste loggade nummer.
unsigned char first_log();	Returnerar index till första loggade nummer.
unsigned char next_log(unsigned char idx);	Itererar ett steg framåt i log-listan.
unsigned char prev_log(unsigned char idx);	Itererar ett steg bakåt i log-listan.
unsigned char remove_log(unsigned char idx);	Tar bort ett nummer från log-listan och returnerar index till föregående log.
unsigned char add_log(unsigned char nbr[]);	Lägger till ett nummer till log-listan och returnerar dess index.
void decode_log(unsigned char idx);	Dekomprimerar en log till en publik vektor.
void encode_log(unsigned char idx);	Komprimerar en log.
unsigned char free_logidx();	Returnerar ett ledigt index. Returnerar 255 om det inte finns något ledigt index tillgängligt.
Meny (för mer utförlig information, se avsnitt 6.2.2)	
void button_handler(unsigned char b);	Hanterar knapptryckningar menytillstånd.
void print_menu();	Skriver ut text som tillhör aktuellt tillstånd till displayen.
unsigned short int dateAddr(int i);	Används för datum- och tidsformatering
unsigned short int dateMin(int i);	Hanterar vilket värde som för tillfället är det lägsta tillåtna för ett visst tecken i datum och tid.
unsigned short int dateMax(int i);	Hanterar vilket värde som för tillfället är det högsta tillåtna för ett visst tecken i datum och tid.
Avbrottsfunktioner	
int exp2();	Funktion som läser av knappar och uppdaterar datum och tid.
int exp5();	Funktion som läser ett tecken från DTMF-kretsen och ser till att systemet hamnar i rätt menytillstånd då ett helt nummer tagits emot.
Övrigt	
void wait(unsigned int n);	Väntar ett antal klockpulser som är proportionellt mot n.

6.2.2 Menysystem

I Appendix finns menysystemets alla tillstånd beskrivna dels med bilder och dels sammanfattade i en tabell. Aktuellt menytillstånd ändras då en knapptryckning registreras i avbrottsfunktionen som tillhör realtidsklockan. Menytillståndet ändras även då DTMF-kretsen har registrerat ett nytt telefonnummer. När ett telefonnummer registrerats visas det på displayen och associerat ljud spelas upp. Efter ett tag återgår systemet till huvudtillståndet.

6.2.3 Komprimering och lagring av loggade nummer

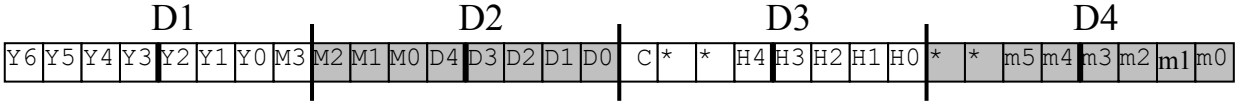
För loggade nummer definierades ett lagringssystem för nummer samt datum och tid i RAM-minnet. Tanken var att minnesarean skulle vara allokerad från början och att det skulle vara enkelt att lägga till och ta bort nya loggade nummer samt bläddra mellan dem samtidigt som de skulle komma i kronologisk ordningsföljd. Dessutom var minnesresurserna relativt begränsade varför det funderades en del över komprimering av data.

En 256*14 byte-matris (logged_nbr[256][14]) används där data finns lagrat enligt tabell 6.3. Två undantag från tabellen finns, nämligen det allra första och det allra sista paketet. Dessa två innehåller aldrig mer än pekare på första, respektive senast inlagda paket vilket innebär att strukturen kan lagra maximalt 254 nummer.

Tabell 6.3 Visar hur loggade nummer sparas undan, Gitlig för i mellan 1 och 254.

`logged_nbr[i][0]`	Pekare till föregående loggat nummer
`logged_nbr[i][1]`	Pekare till efterföljande loggat nummer
`logged_nbr[i][2]`	D1 Datumbyte (definierad nedan)
`logged_nbr[i][3]`	D2 ”
`logged_nbr[i][4]`	D3 ”
`logged_nbr[i][5]`	D4 ”
`logged_nbr[i][6]`	N1 Nummerbyte (definierad nedan)
`logged_nbr[i][7]`	N2 ”
`logged_nbr[i][8]`	N3 ”
`logged_nbr[i][9]`	N4 ”
`logged_nbr[i][10]`	N5 ”
`logged_nbr[i][11]`	N6 ”
`logged_nbr[i][12]`	N7 ”
`logged_nbr[i][13]`	N8 ”

Datumet sparas enligt figur 6.1, där Y står för år, M för månad, D för dag, H för timma och m för minut. Den mest signifikanta biten i D3 (markerad med C i figuren) är en flagga som visar om minnesindexet är ledigt (ej använt eller borttagit) eller ej.

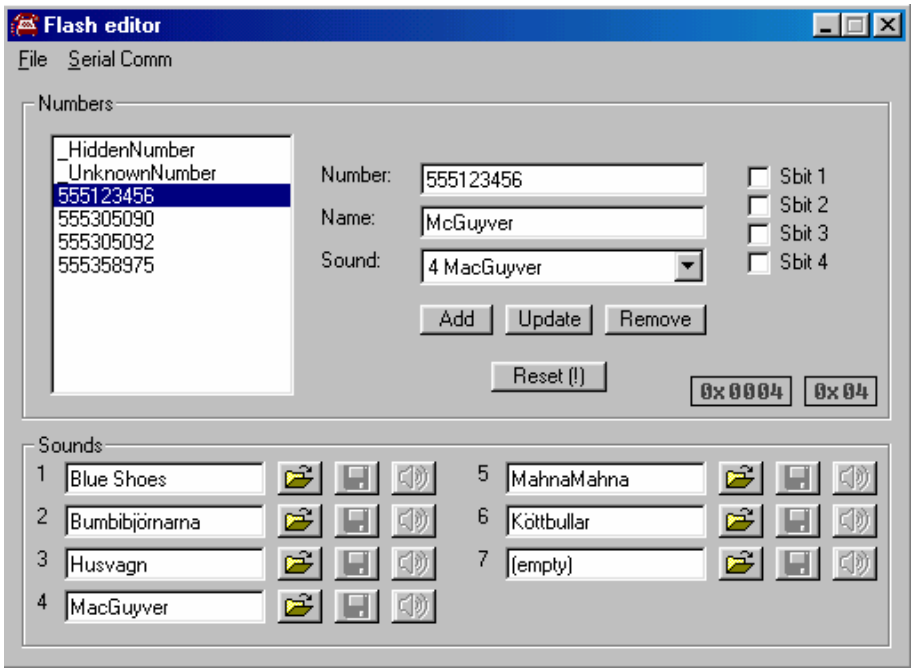


Figur 6.1

När det gäller nummerbytes (N1-N8) är de så definierade att de fyra mest signifikanta bitarna i N1 motsvarar första siffran i numret. De fyra minst signifikanta bitarna i N1 motsvarar andra siffran. Tredje siffran ligger på de fyra mest signifikanta bitarna i N2 och så vidare. För att kunna lagra nummer kortare än 16 tecken definieras alla siffror över 9 (10 i första hand) som mellanslag.

6.3 Flasheditor

Programmet Flash editor är skrivet i Visual Basic 6 och används för att generera/editera binära filer som sedan ska läggas i flash-minneskretsen. Komplett källkod och installationsprogram finns tillgängligt för nedladdning.



Figur 6.2. Flash editor

Till vänster i den övre ramen finns en lista med vilka nummer som finns lagrade i filen. Det finns ett antal speciella nummerposter som börjar med ett ”_”-tecken, vilka motsvarar exempelvis hemliga eller okända nummer. Till varje nummer går det att ställa in vilket namn som ska visas i displayen samt vilket ljud man vill ska spelas upp då ett samtal från numret registreras.

Till höger i den övre ramen finns även fyra checkboxar ”Sbit 1” – ”Sbit 4”. Dessa är de fyra settingsbitarna som är definierade i filstrukturen. Nere till höger i finns två textfält som visar det totala antalet användardefinierade nummer respektive aktuella numrets ”settingsbyte” i hexadecimal notation.

Det finns plats för 7 stycken ljudfiler om 64kB på flashen. I den undre ramen går det importera ljudsampler från standard 8-bitars, 8kHz mono wave-filer. Det går även ange vilket ljudnamn som ska visas i programmet samt på displayen i JukeBox-läge.

På File-menyn går det välja att spara och öppna binära filer. Dialogrutorna som används är windows standard och bör inte kräva närmare beskrivning. För att kunna avgöra om en fil är en giltig flash-fil skrivs ett antal tecken till ett område i början av filen, vilket var definierat som padding i filstrukturen.

7. Resultat och slutsatser

7.1 Påstötta problem

Under projektets gång har vi varit relativt förskonade från hårdvaruproblem. Att alla virningar och lödningar dubbel- och trippelkontrollerats allt eftersom kan nog ha hjälpt. Vi har skyndat långsamt framåt och försökt behandla en krets i taget och inte stressat dit allt på en gång. Helt utan problem har vi ju dock knappast varit.

Den första klockan vi byggde ihop visade sig vara baserat på ett felaktigt kopplingsschema och kristallen svängde med en av de lustigaste sinusvågor vi någonsin skådat.

Många timmar spenderades på att lokalisera underliga problem i mjukvaran, men löste sig när vi upptäckte felet med stackpekarinitieringen i pmain.68k.

Det tog en del tid innan ett problem med avbrott från realtidsklockan var lokaliserat. Vi hade glömt att skicka ack.

7.2 Resultat och utvecklingsmöjligheter

Samtliga krav i den ursprungliga kravspecifikationen har uppfyllts, förutom möjligheten att koppla nummerpresentatören direkt till PC. Vi har haft prototypen igång i 13 timmar i sträck och den verkar ha fungerat utan problem. Det finns vissa smådetaljer i mjukvaran som inte är färdiga, exempelvis en del kod för att kontrollera giltiga datum. Ett annat exempel är att övre gränsen för antal loggade nummer i skrivande stund inte är skyddad, varför programmet kan bete sig underligt om listan blir full.

I sällsynta fall ser det ut som att print-funktionerna kan bli avbrutna av interrupt, vars rutiner i sin tur skriver till displayen. I dessa fall blir det lustiga utskrifter. Detta problem bör gå att lösa genom att avaktivera interrupts temporärt.

När det gäller utvecklingsmöjligheter så är det främst kopplingen till dator som hade varit trevligt att ha. Det är en aning omständigt att behöva plocka ut flash-minnet ur hållaren och programmera den i separat programmerare. Det hade varit betydligt enklare att kunna göra detta direkt på plats.

Det finns som tidigare nämnts i rapporten nästan 1.5kB programminne ledigt och det skulle därmed gå att lägga till ytterliga ett par finesser, exempelvis hälsningsfras och alarmklocka. Dessutom skulle det gå att avvara minne från flashen för att kunna köra små plug-ins.

Vissa kretsar har energisparlägen som vi inte utnyttjar men i en färdig, eventuellt batteridriven konstruktion skulle dessa vara intressanta att titta på.

7.3 Slutsatser

Digitala projekt har varit en mycket lärorik, intressant och kul kurs. Det känns tillfredsställande att ha lyckats bygga något från grunden med hjälp av de erfarenheter man fått under sina år på LTH.

Vi försökte följa förmaningarna om att föra anteckningar under utvecklingsarbetet vilket har varit till stor hjälp vid rapportskrivandet, men även under själva arbetet.

8. Referenser

WAVE File Format, http://www.dazyweblabs.com/wavinfo/

DTMF-signalernas definition, http://www.it.lth.se/digp/DP_dtmf.htm

Bygga oscillator, http://www.it.lth.se/digp/PDF_files/oscillators.pdf

CPU, http://www.it.lth.se/datablad/Processors/68000UM.pdf

EPROM, http://www.it.lth.se/datablad/Memory/eprom/27c64.pdf

RAM, http://www.it.lth.se/datablad/Memory/sram/HM6264.pdf

LCD, http://www.it.lth.se/datablad/display/LCD.pdf

DTMF, http://www.it.lth.se/datablad/periphery/communication/mt8880c.pdf

RTC, http://www.it.lth.se/datablad/Periphery/Rtc/mm58274c.pdf

Latch, http://www.it.lth.se/datablad/Logik/74FCT/74FCT573.pdf

Flash, http://www.it.lth.se/datablad/Memory/flash/am29f040.pdf

D/A, http://www.it.lth.se/datablad/Analog/adda/tlc7524.pdf

OP, http://www.it.lth.se/datablad/Analog/opamp/lm124.pdf

Appendix A Ursprunglig kravspecifikation

Kravspecifikation

- Nummerpresentatör som spelar upp olika ljud beroende på vem som ringer.
- Konfigurerbar med dator i ett användarvänligt grafiskt gränssnitt.
- Tryckknappar för navigering.
- Display som visar senast mottagna nummer.

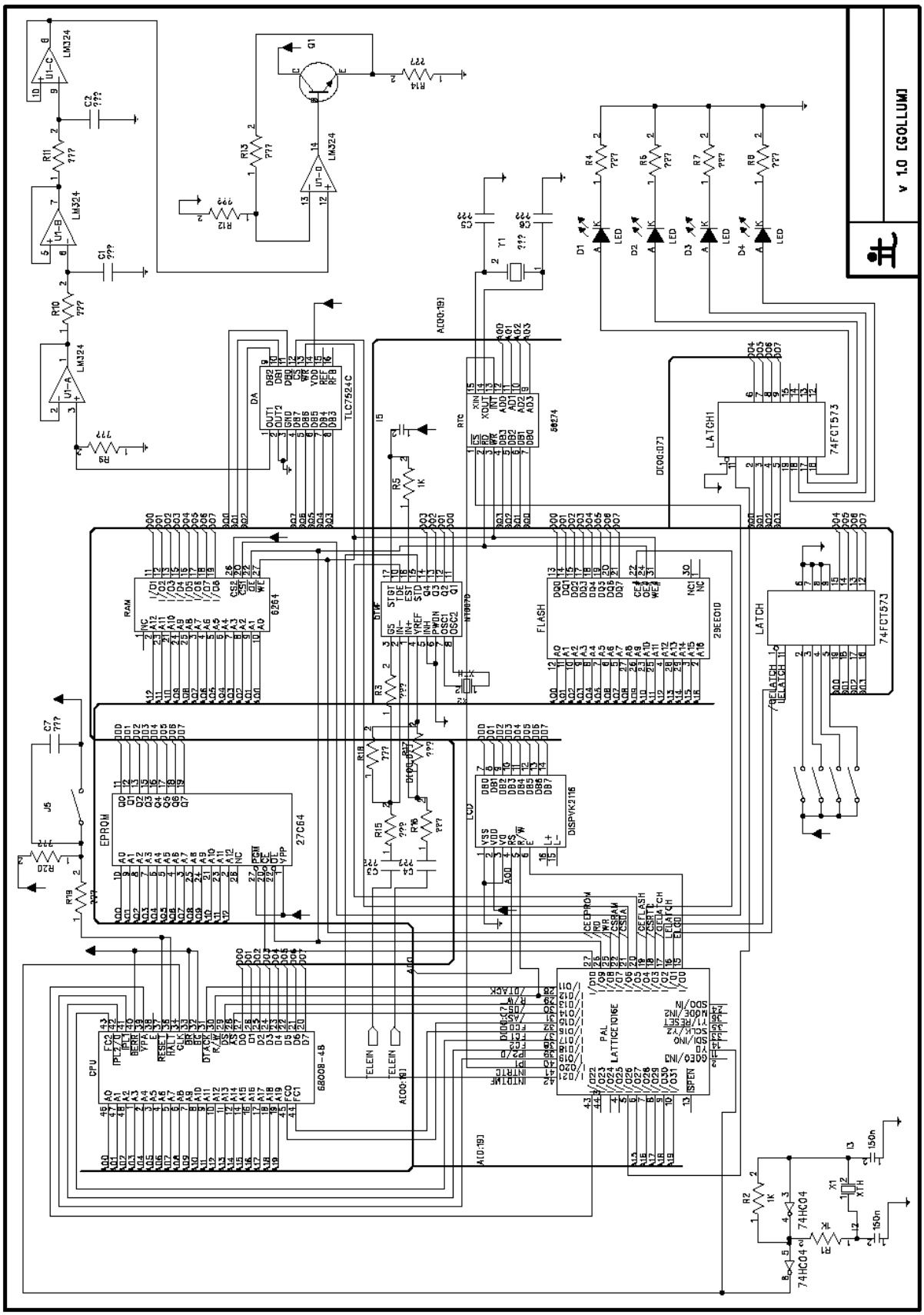
Extra möjligheter

- Digital utgång för anslutning av saftblandare etc
- Möjlighet att ha nummerpresentation i ansluten pc.

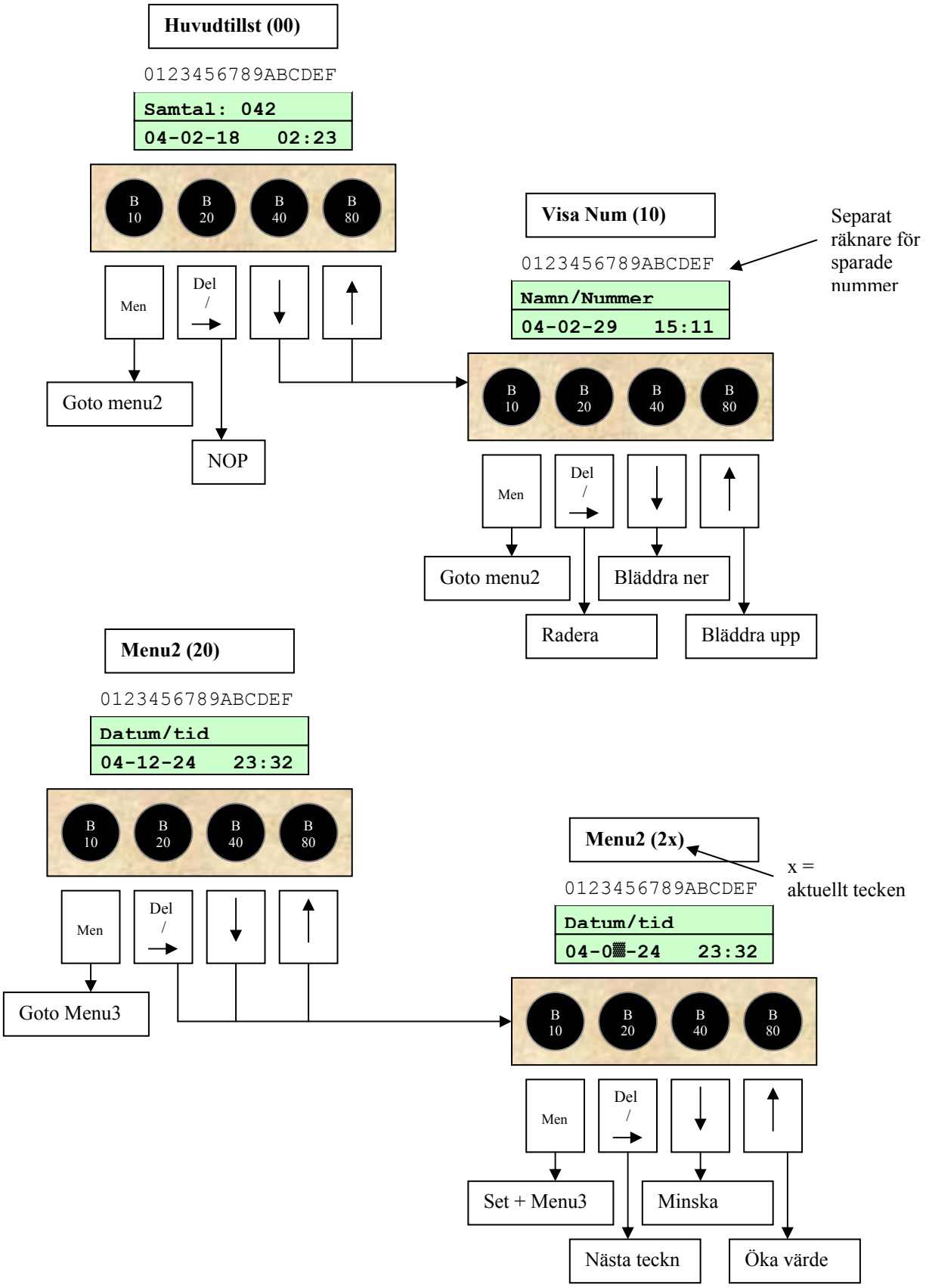
Komponentlista

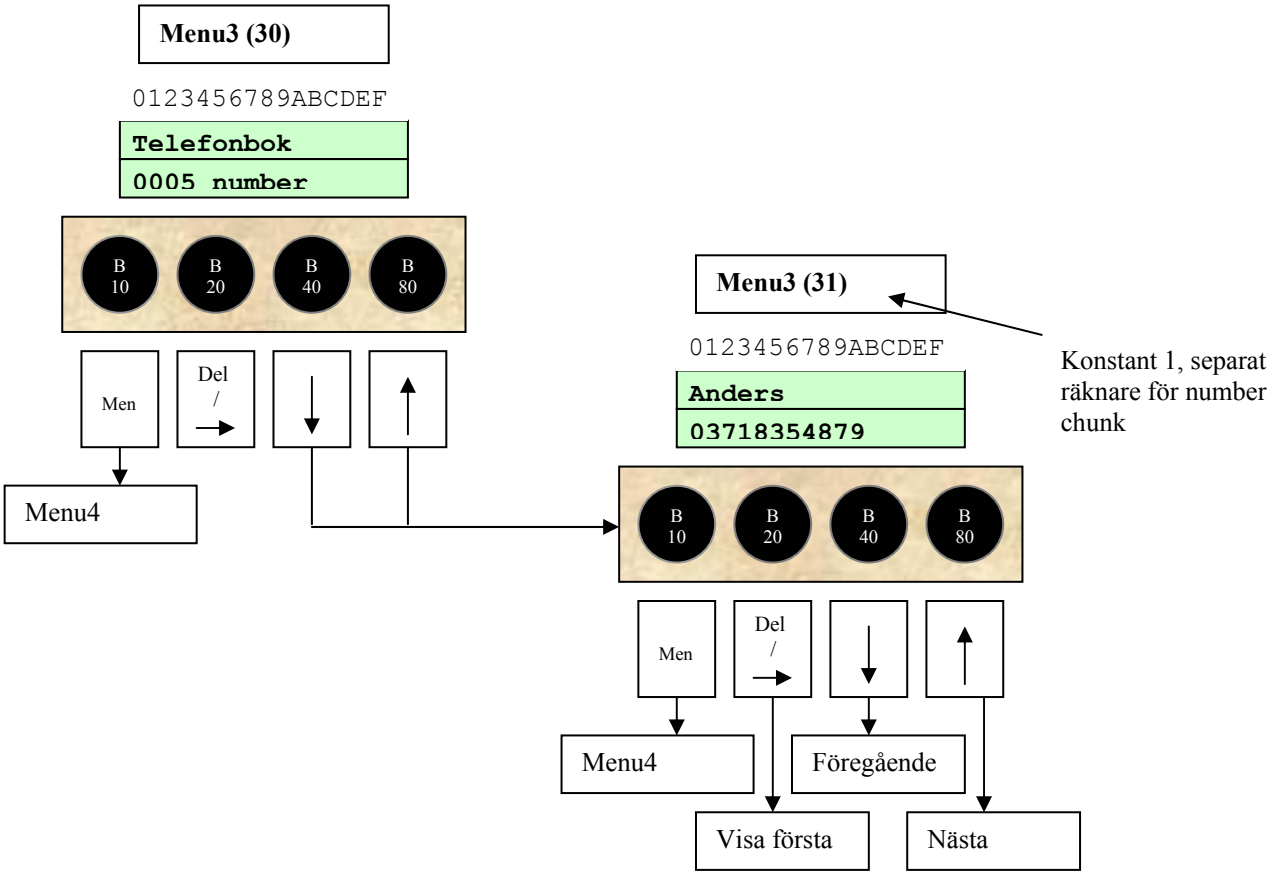
Motorola 68008	
E-prom	Lagra program
EE-prom/flash	Lagra ljud och inställningar
Sram	Arbetsminne
DA-omvandlare	Ljud
Audio amplifier	Ljud
Högtalare	
Programmerbar logik	Adressavkodning
4 tryckknappar	Bläddra, radera, ställa klocka etc
Alfanumerisk teckendisplay	
Serial Communication Controller	
Realtidsklocka	

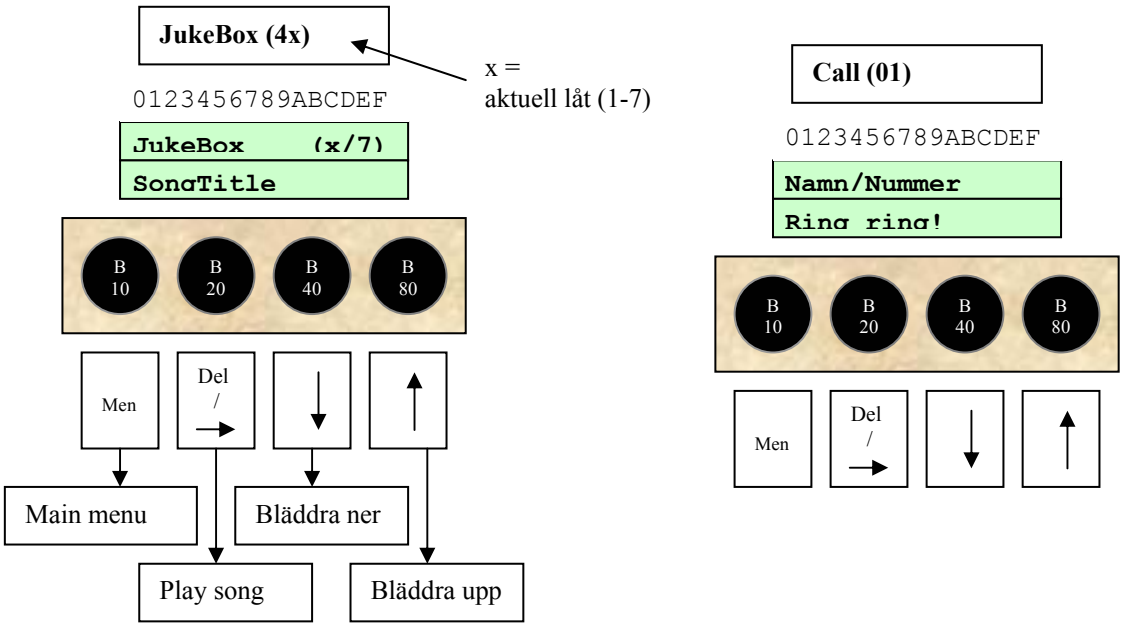
Appendix B Schema



Appendix C Menytillstånd




Menu3 (30)
0123456789ABCDEF
Telefonbok
0005 number
B 10
B 20
B 40
B 80
Men
Del /
Menu4
Menu3 (31)
0123456789ABCDEF
Anders
03718354879
Konstant 1, separat räknare för number chunk
Men
Del /
Menu4
Föregående
Visa första
Nästa

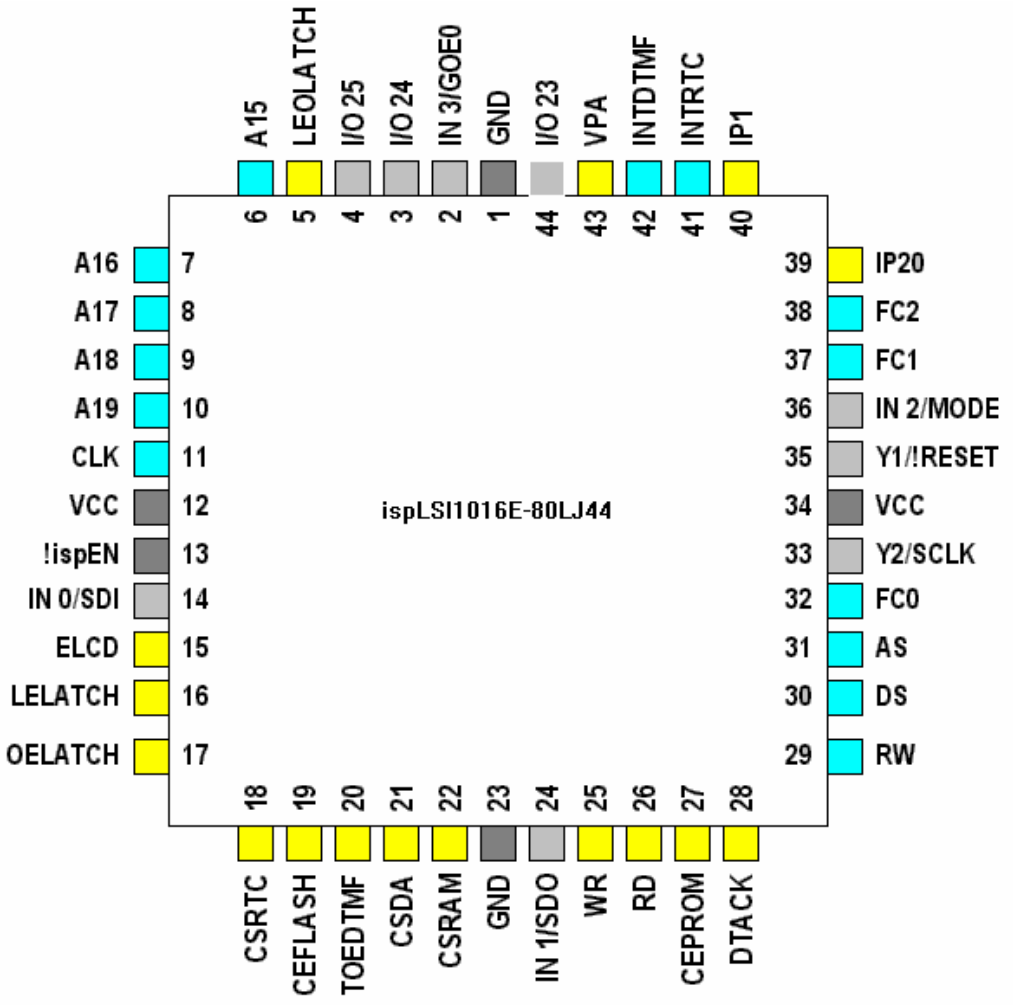

JukeBox (4x)
x = aktuell låt (1-7)
0123456789ABCDEF
JukeBox (x/7)
SongTitle
B 10
B 20
B 40
B 80
Men
Del /
Main menu
Bläddra ner
Play song
Bläddra upp
Call (01)
0123456789ABCDEF
Namn/Nummer
Ring ring!
B 10
B 20
B 40
B 80
Men
Del /

Sammanfattning av menytillstånd

Huvudtillstånd	
00	Visa antal registrerade samtal, datum och tid
01	Menytillstånd som endast uppstår då ett nytt nummer tagits emot. Systemet återgår till tillstånd 00 efter associerat ljud spelats upp.
Meny1 (Visa-meny)	
10	Visa nummer
Meny2 (Datum/Tid)	För att ställa datum och tid. 10 tecken i formatet: YYMMDDhhmm
20	Visa datum och tid utan markör.
21	Markör på år
22	”
23	Markör på månad
24	”
25	Markör på dag
26	”
27	Markör på timma
28	”
29	Markör på minut
2A	”
Meny3 (Telefonbok)	
30	Visa antal nummer
31	Visa ett nummer, separat variabel som håller koll på aktuellt nummer.
Meny4 (JukeBox)	
41	Visa namn på ljudfil 1
42	Visa namn på ljudfil 2
43	Visa namn på ljudfil 3
44	Visa namn på ljudfil 4
45	Visa namn på ljudfil 5
46	Visa namn på ljudfil 6
47	Visa namn på ljudfil 7

Appendix D Lattice

Pin layout



Tabell D.1. Färgkodning av pinar.

	Output pin
	Input pin
	Not used in the program
	Power

Programkod till Lattice-kretsen

```
MODULE Smeagol

TITLE 'Lattice logic in LOTR'

" Inputs
" ------
RW          pin         29; "R/W
!DS         pin         30; "!DS
!AS         pin         31; "!AS

FC0         pin         32;
FC1         pin         37; "Processor status
FC2         pin         38;

A15         pin         6;
A16         pin         7; "Address bus
A17         pin         8;
A18         pin         9;
A19         pin         10;

CLK         pin         11; "Clock

!INTRTC     pin         41; "Interupt from Real Time Clock
INTDTMF     pin         42; "Interupt from DTMF

" Outputs
" -------
ELCD                    pin         15; "Enable LCD
LELATCH                 pin         16; "LE Latch+
LEOLATCH                pin         5;  "LE Output Latch
!OELATCH                pin         17; "!OE Latch
!CSRTC                  pin         18; "!CS Real Time Clock
!CEFLASH                pin         19; "!CE Flash
TOEDTMF                 pin         20; "TOE DTMF
!CSDA                   pin         21; "!CS D/A
!CSRAM                  pin         22; "!CS RAM
!WR                     pin         25; "!Write
!RD                     pin         26; "!Read
!CEPROM                 pin         27; "!CE PROM
!DTACK                  pin         28; "!DTACK CPU

!IP20                   pin         39; "IP2/0 CPU
!IP1                    pin         40; "IP1
!VPA                    pin         43; "VPA"

"States
"------
delay1                  node        istype 'reg';
delay2                  node        istype 'reg';
delay3                  node        istype 'reg';
delay4                  node        istype 'reg';

int50                   node        istype 'reg';
int51                   node        istype 'reg';
```

```
equations

"Read and Write signals
RD = DS & RW;
WR = DS & !RW;

"Delay for DTACK (RTC)
delay1.clk = CLK;
delay1 := CSRTC & !delay1 & !delay2 & !delay3 & !delay4;
delay2 := delay1;
delay2.clk = CLK;
delay3 := delay2;
delay3.clk = CLK;
delay4 := delay3;
delay4.clk = CLK;

"Address decoding

"00000
CEPROM = AS & !A19 & !A18 & !A17 & !A16 & !A15;
"00010

CSRAM = AS & !A19 & !A18 & !A17 & A16 & !A15;
"00100

CSDA = AS & !A19 & !A18 & A17 & !A16 & !A15;
"00101

CSRTC = AS & !A19 & !A18 & A17 & !A16 & A15;
"00110

ELCD  = DS & !A19 & !A18 & A17 & A16 & !A15;
"00111

TOEDTMF = DS & !A19 & !A18 & A17 & A16 & A15;
"01000, Write something (dummy) to the port to enable latch, then read the
result
LELATCH = WR & !A19 & A18 & !A17 & !A16 & !A15;
OELATCH = RD & !A19 & A18 & !A17 & !A16 & !A15;

"01001
LEOLATCH = WR & !A19 & A18 & !A17 & !A16 & A15;

"1xxxx
CEFLASH = AS & A19;

"DTACK
DTACK = CEPROM # CSRAM # CSDA # delay4 # ELCD # TOEDTMF # LELATCH # OELATCH
# LEOLATCH # CEFLASH;

"Interupt decoding

int50.clk = CLK;
int51.clk = CLK;
int50 := IP20 & VPA;
int51 := (int51 & INTDTMF) # int50;

IP20 = INTDTMF & !int51;
IP1 = INTRTC & !IP20 & !int51;
VPA = AS & FC0 & FC1 & FC2;
END
```

Appendix E Programkod

Programkoden finns tillgänglig på *http://www.efd.lth.se/~e00oe/digp/*
eller på Digitala Projekts hemsida.