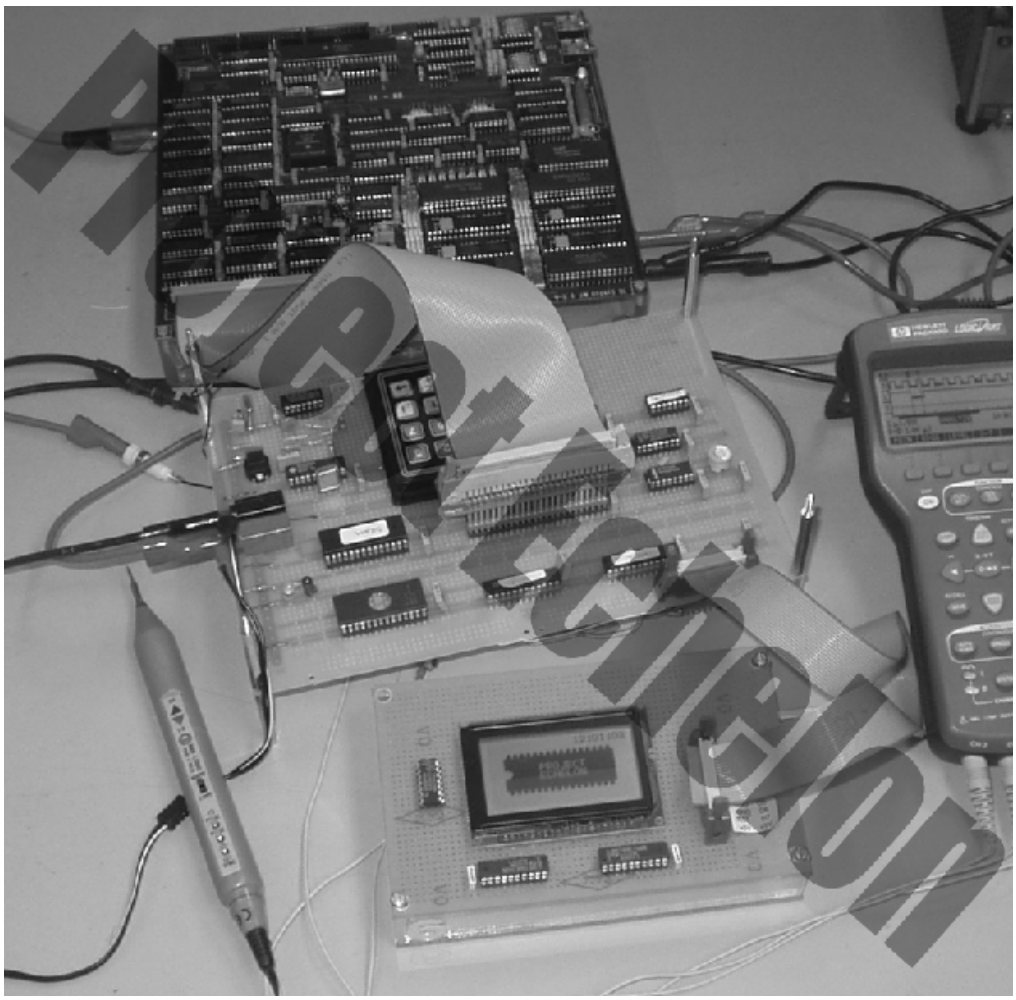
Institutionen för Informationsteknologi
Lunds Tekniska Högskola

Project Echelon

EDI021 Digitala Projekt lp2 HT03



2003-12-09

Joel Olofsson e01
Ulf Nordström e01

Abstract

This document will describe our Project Echelon. Project Echelon is a Caller-ID-machine with graphic display. It has been developed for seven weeks and from the beginning we had some goals about what it was supposed to manage. Most of those goals were reached and many other functions were implemented as the project proceeded. Echelon is now able to show animations, numbers, names and time. To obtain this a Motorola 68008 microprocessor has been used among with a DTMF-tranciever, a realtimeclock, programmable logic, memory and a graphic display. To get everything working some software is needed and ours is written in C. After all work we are proud to present Project Echelon.

Innehåll

1 Inledning

Vi bestämde oss tidigt för att under läsperiod 2 HT 03 läsa kursen Digitala projekt som ges av Institutionen för Informationsteknologi vid Lunds Tekniska Högskola. Den här rapporten kommer att behandla utvecklingen av vårat digitala projekt, nämligen Project Echelon. Allt började med en kravspecifikation och slutade så småningom i ett färdigt system. Utvecklingen kommer att klargöras i rapporten.

2 Kravspecifikationen

Det första steget i detta projekt, som i de flesta andra, var att sätta upp ramar och mål för vad vi ville åstadkomma. Detta resulterade i en kravspecifikation som av praktiska skäl delades in i två delar. Detta då erfarenhet av liknande projekt och deras tidsåtgång saknades. Del ett består av krav som ansågs rimliga att uppfylla under tilldelad tid. Del två består av krav som kan ses som extra funktioner i mån av tid.

2.1 Baskrav

- Visa aktuellt nummer
- Spara 30 nummer med tid och datum
- Lista med nummer som blockeras, telefonen ringer ej, ”Svartlistning”
- Enklare menysystem för inställningar och listor
- Lysdiod visar när någon ringt

2.2 I mån av tid

- Register med namn och telefonnummer, 30 poster
- Visa namn, om det finns med i registret
- Olika ringsignaler, om numret finns med i registret
- Större menysystem för inställningar och listor
- Ring upp aktuellt nummer
- ”Sleep-mode”, utökad ”svartlistning”, endast ett fåtal nummer släpps igenom

3 Nummerpresentation i Sverige

Tilläggstjänsten nummerpresentation tillhandahålls i Sverige av Telia. Den bygger på att det i början av samtalet sänds en kod med information om vem som ringer. Koden sänds med DTMF, Dual Tone Multiple Frequency. Som namnet antyder sänds två frekvenser samtidigt enligt tabell 3.1. Tonerna sänds i 70 ms långa pulser med 70 ms mellanrum. Först sänds en startbit följt av numret och slutligen sänds en stoppbit.

f/Hz	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

Tabell 3.1 DTMF-tonernas frekvens-avkodning

4 Konstruktionsarbetet

Konstruktionsarbetet kan delas upp i två delar, hårdvara och mjukvara. Första steget var valet av hårdvara vilket beskrivs nedan. För att ha en skiss att arbeta efter ritades ett schema i Powerlogic. När det var avklarat skulle komponenternas socklar placeras på virkortet och matningsspänning samt jordning lödas på plats. Avkopplingskondensatorer sattes i anslutning till alla kretsar för att ta upp eventuella störningar. Efter lödningen var avklarad virades övriga ledningar enligt schemat. Även ett jordnät virades för att minska störningar.

Nu var det dags att verifiera att hårdvaran fungerade som det var tänkt. Först programmerades logiken för att styra adressering av kretsarna. De fem högsta adressbitarna används för att bestämma vilken komponent som ska ha tillgång till adress- och databussen. Kodningen vi använde oss av kan ses i figur 4.1. Logiken styrde även när processorn skulle gå över och jobba i synkront läge. Programmeringen kan ses i Appendix A och B. Med logiken programmerad var det dags att testa komponenterna en i taget med hjälp av utvecklingssystemet it68.

A19	A18	A17	A16	A15	HEX	Krets
0	0	0	0	*	0	EPROM
0	0	0	1	0	10000	SRAM
1	0	0	1	1	98000	LCD1 data
1	0	0	1	0	90000	LCD1 instruction
1	0	1	0	1	A8000	LCD2 data
1	0	1	0	0	A0000	LCD2 instructions
0	1	0	0	0	40000	RESETLCD
0	0	1	1	1	38000	DTMF RS0=1
0	0	1	1	0	30000	DTMF RSO=0
0	1	1	1	0	70000	REAL_CLOCK
1	1	1	1	0	F0000	TANG_DECODE

Figur 4.1 Adresskodning

it68 är ett utvecklingssystem som emulerar processorn. Utvecklingssystemet är kopplat till en dator via seriellporten och en hyperterminal används för att skicka kommandon. Med hjälp av läsningar och skrivningar till olika adresser kan de olika kretsarna testas. När sedan mjukvaran skall testköras kan brytpunkter införas och underlätta felsökning.

Med utvecklingssystemet sammankopplat med konstruktionen började det stora felsökningsarbetet, mestadels med logikprob men även med logikanalysator. Tack vare att kretsarna testades en i taget var det oftast relativt enkelt att identifiera vilken krets som orsakade felen.

Efter många timmars testning, felsökning och omvirning var det hög tid att gå över till mjukvaran. Mjukvaran är skriven i C samt vissa delar i assembler. Assemblerkoden som användes var redan färdigskriven och endast ett par ändringar utfördes. Mer om mjukvaran kan läsas i kapitel 4.2. När mjukvaran var färdig och fungerade tillfredställande kunde EPROM:et brännas och konstruktion köras fristående från utvecklingssystemet.

4.1 Hårdvara

Hårdvaran är den fysiska delen av projektet där alla komponenter ska väljas och kopplas samman. Processorn är hjärtat i konstruktionen och vi hade två processorer att välja mellan, Motorola HC11 och Motorola MC68008. HC11 är en enchipsdator med inbyggda funktioner som minskar mängden övrig hårdvara. MC68008 är en ren processor där man får lägga till hårdvara efter eget behov. Eftersom MC68008 ger en bättre bild av hur en processor arbetar valdes denna till projektet.

Alla komponenter som använder sig av databuss och adressbuss kan ställa dessa in- och utgångar i tristate för att inte störa andra komponenter. Tristate styrs av ”chip-select”, en signal som logiken genererar, som talar om vilken komponent som får använda adress- eller databussen.

4.1.1 Komponentlista

- Processor, Motorola MC68008
- Programmerbar logik, Lattice GAL22V10, två stycken
- 32 kb EPROM, Fairchild NM27C256
- 32 kb SRAM, NEC μ PD43256B
- LCD, Batron BT 128064 B
- Realtidsklocka, National Semiconductor MM 58274 C
- DTMF-tranciever, MITEL MT8880C
- 16 knappars tangentbord
- Tangentbordsavkodare, National Semiconductor MM54C922
- JK-vippa, Texas Instruments CD74HC73
- 16 bitars räknare, Texas Instruments SN74HC163
- 8 MHz oscillator

4.1.2 Motorola MC68008

MC68008 är en vidareutveckling av MC68000 med 22 bitars adressbuss och 8 bitars databuss vilket möjliggör adressering av upp till 4Mb minne. Detta gäller för 52 pinnars modellen men vi använder oss av en 48 pinnars modell som har två färre adressbitar vilket resulterar i att den kan adressera 1Mb minne. Processorn har åtta 32 bitars dataregister, sju 32 bitars adressregister och en stackpekare. Utöver detta finns även ett 16 bitars

statusregister och en 32 bitars programräknare. Processorn arbetar vid en klockfrekvens upp till 10 MHz och det finns två avbrottsingångar för externa avbrott. Processorn arbetar normalt asynkront men med externa kretsar kan man även få den att arbeta synkront. Detta behövdes t ex vid kommunikation med LCD-displayen och DTMF-trancievern i denna konstruktion.

4.1.3 Programmerbar logik

För styrning av externa kretsar och avbrott behövs någon form av logik. Fördelarna med programmerbar logik är att den tar liten plats och det är enkelt att genomföra ändringar. Vi valde att använda oss av två stycken Lattice GAL22V10. Det är en krets med 10 ingångar och 12 valbara in- eller utgångar. Det finns även möjlighet att klocka kretsen så att vippor kan implementeras på utgångarna.

4.1.4 EPROM

I EPROM:et lagras den kod som processorn ska exekvera. Ett ganska stort EPROM valdes, 32 kb, för att vara säkra på att få plats med all programkod. Detta visade sig vara ett klokt val då 8 kb snabbt överskreds och mot slutet även 16 kb. EPROM:et använder sig av en 15 bitars adressbuss och en 8 bitars databuss.

4.1.5 SRAM

I SRAM:et skall all temporär data lagras under exekveringen. Eftersom det i förhand är svårt att veta hur mycket utrymme som behövs så valdes även här ett relativt stort minne. Temporär data som lagras är t ex variabler och stacken. SRAM:et har precis som EPROM:et en 15 bitars adressbuss och 8 bitars databuss.

4.1.6 LCD

Batron BT 128064 B är en grafisk LCD med en upplösning på 128x64 pixlar. Displayen arbetar synkront och detta innebar att en konstruktion med JK-vippor fick användas för att få processorn att också arbeta synkront när den kommunicerade med displayen. Displayens drivkretsar använder sig av en databuss på 8 bitar för instruktioner och data.

4.1.7 Realtidsklocka

Från början valdes en RTC58321 från Epson. Den visade sig senare inte fungera med våran processor utan behövde IO-interface för att kunna användas. Istället valdes MM 58274 C från National Semiconductor vilket visade sig fungera betydligt bättre. Klockkretsen drevs av en kristall på 32,768 kHz kopplad med två kondensatorer varav den ena var ställbar och används för kalibrering. Realtidsklockan har en 4 bitars adressbuss och en lika stor databuss.

4.1.8 DTMF-tranciever

DTMF-trancievern tar emot DTMF-tonerna från telenätet och avkodar dem för att lägga ut datan på databussen. DTMF-trancievern kan även sända DTMF-toner vilket kan användas för uppringning av ett mottaget nummer. Det var tänkt att denna funktion skulle implementeras i mån av tid, detta inträffade dock inte.
DTMF-trancievern var enkel att använda i konstruktionen eftersom den själv genererar avbrottssignal när DTMF-toner kommer. För kommunikation använder den sig av en 4 bitars datorbuss. DTMF-trancievern arbetar synkront precis som displayen och samma teknik fick användas även här.

4.1.9 Tangentbord och avkodare

Ett 16-knappars tangentbord användes tillsammans med en avkodare från National Semiconductor, MM54C922, för inmatning av information. Endast sex stycken av knapparna användes eftersom sifferknapparna känns överflödiga vid jämförelse med liknande kommersiella produkter. Avkodaren använder en 4 bitars datorbuss och har en signal som säger till när data finns att hämta, d v s när någon tryckt på en knapp.

4.1.10 Övriga komponenter

De två JK-vipporna användes för att få processorn att gå över och arbeta synkront vid kommunikation med displayen. Konstruktion kan ses i kopplingsschemat i appendix C.

Räknaren användes för att adressbussen skulle hålla kvar adresserna då realtidsklockan behövde kunna läsa dem i minst 390 ns.

Oscillatorn konstruerades kring en 8 MHz kristall vilken kan ses i kopplingsschemat.

4.2 Mjukvara

Från mjukvaran kommer alla kretsar att styras. DTMF-avkodaren och tangentbordet är kopplade så att de genererar avbrott. Grundidén med C-programmet är att polla realtidsklockan och när avbrott från DTMF-avkodaren eller tangentbordet kommer tas de omhand och åtgärder vidtas. Då en grafisk display valdes som saknade teckentabell fick en sådan implementeras. En grafisk display gjorde mjukvaruarbetet betydligt jobbigare men öppnade samtidigt för fler möjligheter som bilder och animerade sekvenser.

Ett menysystem implementerades med val för de funktioner som var nödvändiga och senare tillkom även ett mindre nödvändigt alternativ, ”Mata pacman”, vilket innebar att 16 Pacman’s äter upp samtliga menyer på displayen. De funktioner som är skrivna behandlas i nedanstående stycken.

4.2.1 Display

li(unsigned short int hex)
ri(unsigned short int hex)

Läser statusregistret från displayen och kontrollerar att inte displayen är upptagen. När ”busy”-flaggan är låg skickas instruktionen *hex* till displayen. *li* skickar instruktionen till vänstra display-halvan och *ri* till högra.

ld(unsigned short int hex)
rd(unsigned short int hex)

Läser statusregistret från displayen och kontrollerar att inte displayen är upptagen. När ”busy”-flaggan är låg skickas datan *hex* till displayen. *ld* skickar data till vänstra display-halvan och *rd* till högra.

clearDisp()

Rensar hela displayen.

clearRow(unsigned short int row)

Rensar raden *row*, 0-7.

4.2.2 Tangentbordet

exp2()

Avbrottsrutinen för tangentbordsavkodaren. Läser in 8 bitar från avkodaren och maskar ut de 4 sista och sparar dem i den globala variabeln *key*. Ändrar även en global variabel *change* till 1 om den är 0 och 0 om den är 1. Detta för att enkelt kunna se om en ny knapptryckning inträffat.

4.2.3 DTMF

exp5()

Avbrottsrutinen för DTMF-trancievern. Körs varje gång en ton kommer. För en sekvens toner när ett samtal kommer sparas hela sekvensen och klockslaget i de globala två-dimensionella vektorerna *allNbr[][]* och *allNbrTime[][]*, startbit och stoppbit sorteras bort. De globala variablerna *nbrOfCalls* och *newNbrs* ökas.

4.2.4 Realtidsklockan

updateClock()

Uppdaterar klockan om tiden ändrat sig sedan den sist kördes. Skriver ut klockan uppe i högra hörnet av displayen.

updateClockFast()

Används vid inställning av tid i menyvalet ”STÄLL KLOCKAN”. Uppdaterar klockan utan att skriva ut den på displayen.

4.2.5 Menysystemet

mainMenu()

Menyalternativen glider in, ett i taget, från högra kanten. Med hjälp av upp och ned knapparna samt enter-knappen kan menyval göras.

menuRecieved()

Visar de 50 senast inkomna numren eller namn.

menuAddName()

Registrerar ett namn med tillhörande nummer. Registret har plats för 50 namn/nummer.

menuClock()

Ställer klockan.

menuShow()

Visar registrerade namn med tillhörande nummer.

menuPacman()

Pacman med vänner kommer in från sidorna och äter upp allt i sin väg.

4.2.6 Övriga funktioner

putChar(char ch, int c, int r, int hl)

Letar igenom teckentabellen och ser om tecknet *ch* finns med, annars skrivs ett blanksteg ut. *c* anger hur många pixlar, 0-127, från vänsterkanten tecknet ska skrivas ut. *r* anger raden, 0-7. Om *hl* = 0 skrivs tecknet med fyllda pixlar, om *hl* = 1 inverteras tecknet. Varje tecken är 6x8 pixlar stort.

str(char st[], int col, int row, int hl)

Skriver ut strängen *st[]* vid positionen som bestämts av *col* och *row*. *col* räknas här i 20 steg per rad, ett steg per tecken. *row* anger raden, 0-7. *hl* har samma funktion som ovan fast för hela textsträngen.

slidingString(char st[], int startCol, int endCol, int row)

Strängen *st[]* glider från position *startCol* till position *endCol* på raden *row*. Positionsvariablerna räknas som ovan.

dispLatestNbr()

Skriver ut numret senast mottagna numret på displayen. Om numret finns med i registret skrivs istället tillhörande namn ut.

present()

Intro som körs när systemet startas.

env(unsigned short int new)

Ett kuvert visas med en text som talar om hur många nummer som ringt sedan man senast läste av numren.

dispNbr(unsigned short int nbr[20])

Visar nummer, eller namn om det finns i registret, samt en animation när det ringer.

5 Resultat och diskussion

När vi nu ser tillbaks på kravspecifikationen har vi lyckats uppfylla alla baskrav utom svartlistning. Detta krav ströks ganska tidigt då det krävde ytterligare hårdvarukonstruktion och vi var osäkra på om tiden skulle räcka till för mjukvaran. Av kraven som skulle uppfyllas i mån av tid kunde hälften uppfyllas vilket vi är nöjda med.

Med den kunskap vi nu besitter kan man vara efterklok på ett par punkter. Dels skulle den programmerbara logiken ha varit färdigskriven tidigare och dels skulle timingen ha kontrollerats på samtliga komponenter. I det stora hela är vi mycket nöjda med både kursen och konstruktionen.

6 Referensförteckning

[1] “Motorola Semiconductors Advance Information MC68008”, April 1985
[2] ”it-68 Utvecklingssystem för MC68008 Version 4.0”
[3] R. Gandvik & G. Engström, “Description of network interfaces; Analogue access to PSTN”, Telia, September 2000
[4] Mitel MT8880C Integrated DTMF Transceiver Datasheet.
[5] Fairchild NM27C256 EPROM Datasheet
[6] NEC μ PD43256B SRAM Datasheet
[7] Batron BT 128064 B Datasheet
[8] National Semiconductor MM 58274 C Datasheet
[9] National Semiconductor MM54C922 Datasheet
[10] Texas Instruments SN74HC163 Datasheet

Appendix A

```
'GAL 1
device 22V10

VPAFF           1       'VPA från synkroniseringskrets
A15             2       'adressbit 15
A16             3       'adressbit 16
A17             4       'adressbit 17
A18             5       'adressbit 18
A19             6       'adressbit 19
FC0             7       'processorns statusregister
FC1             8       '----------||--------------
FC2             9       '----------||--------------
AS              10      'adress-strobe från CPU
COUNT           11      'fördröjd dtack
GND             12      'jord
VMA             13      'VMA från synkroniseringskrets
VPACPU          14      'VPA till CPU
CEEPROM         15      'chip-select EPROM
CSSRAM          16      'chip-select SRAM
CSREAL          17      'chip-select Realtidsklockan
CSDTMF          18      'chip-select DTMF-tranciever
OETANG          19      'output-enable tangentbordet
DTACK           20      'data transfer acknowledge till
CPUVPALCDDTMF   21      'gå över till synkronläge
CS1LCD          22      'chip-select LCD vänster
CS2LCD          23      'chip-select LCD höger
VCC             24      'matningspänning

start

CEEPROM         /= /A15*/A16*/A17*/A18*/A19*/AS;
CSSRAM          /= /A15* A16*/A17*/A18*/A19*/AS;
CSDTMF          /=  A16* A17*/A18*/A19*/AS*/VMA;
OETANG          /= /A15* A16* A17* A18* A19*/AS;
DTACK           /= /CEEPROM + /CSSRAM + /OETANG +
                COUNT;
CS1LCD          /=  A16*/A17*/A18* A19*/AS*/VMA;
CS2LCD          /= /A16* A17*/A18* A19*/AS*/VMA;
VPALCDDTMF       =  A16*/A17*/A18* A19*/AS+/A16*
                A17*/A18* A19*/AS+
                /A15*/A16*/A17*A18*/A19*/AS +
               A16* A17*/A18*/A19*/AS;
VPACPU          /= /VPAFF + FC0* FC1* FC2*/AS;
CSREAL          /=  /A15* A16* A17* A18*/A19*/AS;
end
```

Appendix B

```
'GAL 2
device 22V10

DAVTANG         1       'data available
A15             2       'adressbit 15
A16             3       'adressbit 16
A17             4       'adressbit 17
A18             5       'adressbit 18
A19             6       'adressbit 19
VMA             7       'VMA från synkroniseringskrets
OETANG          8       'output-enable tangentbordet
Rw              9       'RW från CPU
AS              10      'adress-strobe från CPU
DS              11      'data-strobe från CPU
GND             12      'jord
INTDTMF         13      'avbrott från DTMF
DILCD           14      'DI till LCD
RESETLCD        15      'reset till LCD
RSODTMF         16      'RSO till DTMF
BEGIN           17      'starta fördröjning av dtack
OE              18      'output-enable
WE              19      'write-enable
LIGHT           20      'signal till LED
NC4             21      'no connection
NC5             22      'no connection
IPL1            23      'avbrott från tangentbord till CPU
VCC             24      'matningsspänning

start

OE              /=  RW */DS;
WE              /= /RW */DS;
DILCD           =  A15* A16*/A17*/A18* A19*/AS*/VMA +
                A15*/A16* A17*/A18* A19*/AS*/VMA;
RESETLCD        /= /A15*/A16*/A17* A18* /A19*/AS*/VMA;
RSODTMF          =  A15* A16* A17*/A18*/A19*/AS;
ARESET          = /OETANG;
LIGHT           /= INTDTMF;
IPL1            /= 1;
BEGIN           =  /A15* A16* A17* A18*/A19*/AS;

end
```