LUNDS TEKNISKA HÖGSKOLA
Lunds universitet

Projekt Rapport

Datum: **2002-12-05**

Utfört av: **André Månsson E99**
Henrik Henriksson E99

Handledare: **Bertil Lindvall**

André Månsson E99
Henrik Henriksson E99

Projektbeskrivning

Vi har valt att konstruera en väderstation med inbyggd datalog.
Konstruktionen bygger på den välkända processorn MC68008 från Motorola.

Konstruktionen har döpts till ***SUNe***. Denna väderstation loggar yttertemperatur, vindstyrka samt vindriktning. Uppmätta data representeras på två olika displayer. En alfanumeriskdisplay som har till uppgift att representera inmatning och data i form av siffror och bokstäver, en grafisk display för att visa grafer och trender. Inmatningen till systemet sker via en 16-knappars knappsats.

Konstruktionsmetoden har byggt på virning och viss lödning. Systemet har sedan testats med hjälp av ett utvecklingsverktyg för MC68008 som institutionen har tillhandahållit.

André Månsson E99
Henrik Henriksson E99

Innehåll

Appendix:

Inledning

Vi läser en kurs som heter Digitala Projekt på IT-institutionen. I denna kurs skall man konstruera en applikation med hjälp av digitala komponenter. Vi har valt att bygga en väderstation med inbyggd datalog.

Specifikationerna för projektet som vi satte upp då vi började är följande:

1. Vi skall kunna logga yttertemperatur och vindstyrka (ev. luftfuktighet och lufttryck) samt representera vindriktning.
2. Vi skall kunna representera aktuella värden på en display och representera grafer och trender på en annan.
3. Man skall kunna hämta upp gamla ”aktuella” värden ett år tillbaka i tiden med hjälp av att mata in ett datum.
4. Lägga stor vikt vid användarvänlighet.

Projektet har vi valt att bygga kring den välkända processorn MC68008 från Motorola. Till vår hjälp har vi haft ett mycket bra utvecklingsverktyg som institutionen har tillhandahållit. Själva hårdvaran bygger på standardkomponenter såsom 74-serien men även programmerbara logiska kretsar så som PAL22V10.

Förbindningarna mellan de olika kretsarna har skett med hjälp av virning.
Det är endast spänningsmatning samt vissa diskreta komponenter som lödats på vårt experimentkort.

Hårdvara

Hjälpen vi fått vid hårdvarukonstruktionen är från de datablad som finns publicerade på kursens hemsida samt vissa tips från vår handledare.

Processor: 68008
Detta är en processor från Motorola med asynkron databuss som är släkt med 68000.
Den har 48 pinnar. Varav 20 pinnar är adresspinnar och 8 är datapinnar.
Vilket gör att vi kan adressera upp till 1 MByte av minne.

Klocka:
Systemet klockas av en central klocka på 10 MHz. Till detta använder vi en integrerad oscillator vid namn EXO 3. Mycket kompetent oscillator på 20 MHz som delas ner genom att man anger en 3 bitars kombination. Grundfrekvensen kan delas ner från 2 till 256.

ROM-Minne: 27C64
Är ett 8 KByte ROM-minne som vi lagrar den kompilerade C-kod som utgör mjukvaran för vårt system.

RAM-Minne: 431000
Är ett 128 KByte stort RAM-minne som lagrar loggade värden samt de temporära variabler som mjukvaran behöver. Detta är ett ganska väl tilltaget minne, detta på grund av att vi ville kunna ha möjligheten att logga fler parametrar än de vi startade med.

PAL: PAL22V10
Vi använder tre stycken sådana här programmerbara kretsar i vår konstruktion. De två första används primärt för att kunna koda ut ”chip select” ur adresserna givna från processorn dvs. mjukvaran. Den sista används för att kunna styra de tre olika interrupt som vi har. De tre olika interrupten som vi använder oss av är från knappsats-decodern, A/D-omvandlaren samt realtidsklockan. Programkoden till PAL:arna är bifogade i appendix A.

Knappsats avkodare: 74C992
Detta är en standard 16-key decoder för vår knappsats.
Knappsatsen fungerar så att den skickar ut vilken knapp som var nertryckt i form av att ange vilken kolumn och rad knappen satt på.
Decoderns uppgift är sen att skicka ut en 4 bitars datasträng på data bussen samt sätta en flagga ”data available”. Det är sedan denna flagga som vi använder till PAL:en för att ange ett interrupt. För att interrupten skall bli i form av en puls och inte en kontinuerlig signal är vi tvungna att använda en 74LS74 där ”data available” är kopplad på klockan på FlipFlop:en. För att man inte skall få dubbla knapptryckningar på knappsatsen (sk. Key Bounce) så kopplar man in två kondensatorer på två ingångar på den integrerade decodern.

Givare:
De givare vi hittills använt oss av är en LM335 temperaturgivare samt en kombinerad vindhastighets- och vindriktningsgivare.
Vindriktiningsgivaren är helt enkelt en potentiometer, och vindhastighetsgivaren är en snurra med radien på 5cm som ger en puls per varv.
Tempgivaren LM335 är en linjär givare som ger ut 10 mV/K.

A/D-Omvandlare: ADC0809
Detta är en 8 bitars A/D-omvandlare med 8 adresserbara kanaler.
Med möjligheten att sätta referensspänningar. A/D-omvandlaren startar sin omvandling genom att vi adresserar den i vår mjukvara. Då vi adresserat A/D:n skickar PAL:en en signal ”AD Start” som startar omvandlingen. Då A/D:n är klar skickar den en interruptsignal i form av en ”End Of Conversion”. På samma sätt som knappsatsavkodaren måste vi omforma interruptsignalen från en kontinuerlig signal till en pulssignal, detta gör vi med en 74LS74 (Dual FlipFlop).
Till A/D-omvandlaren kopplar vi in tempgivaren samt vindriktningsgivaren, då den har ytterligare 6 kanaler finns det god chans att utveckla konstruktionen med ytterligare givare så som tryck och fuktighet.

Pulsräknare: 74HC590
Detta är en 8 bitars pulsräknare. Denna använder vi för att räkna pulserna från vindhastigshetsgivaren. Den ger en puls per varv. Här stötte vi på vårt första problem. Räknaren räknade helt galet. Då vi tittade på pulsen i ett oscilloskop såg vi att pulsens form var helt konstig, vi hade övertoner och oscillationer. Detta försökte vi avhjälpa med hjälp av att lägga en kondensator parallellt över givaren, detta hjälpte en del, men ej tillräckligt. Vi var tvungna att lägga in en form av ”puls shaper” för att få pulsen fyrkantig och fin. Detta gjordes med hjälp av en 74HC14 SchmittTrigger.

Realtidsklocka: ICM7170
Detta är en realtidsklocka med inbyggt larm (larmen använder vi inte).
Denna klocka räknar upp år, månader, dagar osv. ända ner till 100 dels sekunder. Den är ställbar genom att adressera och skriva till ett speciellt register.
Vi stötte på ett litet problem i början med denna klocka, då vi ställt in en tid och sedan skulle läsa ut den fick vi fortfarande det gamla värdet. Detta förklarades sedan genom att registret vi skrev till var latchat. Dvs. den data som vi skrev kom aldrig igenom. Detta löses genom att man alltid läser adressen för 100-dels sekunder, får då latchar registret igenom datan.
Denna RTC kräver sin egen klocka, vilket vi byggde genom att använda en 32 KHz kristall och två kondensatorer varav en är variabel.

Alfanumerisk display:
Detta är en vanlig 4x16 alfanumeriskdisplay, vilket betyder att den har 4 rader med möjligheten att representera 16 tecken per rad. Fördelen med en sådan här display är att den kan representera ASCII-tecken. För att få rätt kontrast på denna fyraraders display så var man tvungen att använda en negativ spänning. Då kortet som vi använder endast matas med +5 V så fick vi använda en DC/DC konverter. Konvertern vi använde oss av heter MAX635. Denna konverter omvandlar +5V till -5V. Då endast krävde ca -1 V för att displayen skulle fungera så spänningsdelade vi via en potentiometer på 10 kΩ

Grafisk display:
Detta är en 64x128 grafisk display. Denna display använder vi för att kunna representera grafer och trender. Denna display var ganska svår att förstå sig på samt att en bild tog ganska mycket minne (1 bild = 1KByte).

För att driva de båda displayerna krävs en synkron buss. Då vi använder processorn 68008 som har en asynkron buss var vi tvungna att omforma denna till en synkron då vi använder displayerna. Detta gjordes med hjälp av en 74HC73 som är en J/K-vippa.

Kopplingsschema har framställts med hjälp av PowerLogic och finns i appendix.

Enkel manual till SUNe

Då man startar SUNe hamnar man i vad vi kallar ”main screen”. Denna skärmbild visar aktuell tid och datum samt aktuell vindstyrka, vindriktning samt temperatur.

A. För att komma vidare så måste man trycka ”Enter”
B. Då man tryckt ”Enter” hamnar man i meny 2. Här kan du välja mellan tre olika alternativ genom att trycka på 1, 2 eller 3.
   1. ”Choose Graph”
   2. ”Get Data”
   3. ”Setup”
C. Väljer man 1 dvs. ”Choose Graph” får man ytterligare tre alternativ
   1. ”Year Average”
   2. ”Month Average”
   3. ”Day Average”
D. Vad man än väljer i denna meny så får man frågan om man vill se temperatur eller vindstyrka. Då man valt något av dessa så kommer en trend visas på den grafiska displayen.
E. Väljer man istället ”Get data” i meny 2 så hamnar man i en skärmbild där man skall ange ett datum. Då man angett ett giltigt datum och tryckt ”Enter” så kommer displayen visa alla relevanta medelvärden från just den dagen. Skärmen kommer att återgå till ”main screen” efter 2 min.
F. ”Setup” i meny är till för att ställa in datum och tid samt ange vilken riktning som vindriktningsvisaren sitter.

Finns alltid möjlighet att avbryta det man håller på med genom att trycka på ”Main Menu”. Vill man gå tillbaka till föregående meny trycker man på knappen ”Back”.

Mjukvara:

Det första man måste bestämma sig för är i vilka minnes areor som de olika adresserbara kretsarna skall ligga. Vi har valt följande mapp.

EPROM:	00000 (Rom-minnet)
RTC:	80000 (Realtidsklockan)
ADCW:	0A000 (A/D Start)
ADCR:	0B000 (A/D Läsa)
KEY:	0C000 (Keydecodern)
CNT:	0F000 (Räknaren)
GDISP1:	11000 (Första 64x64 segmenten i grafiska displayen)
GDISP2:	13000 (Andra 64x64 segmenten i grafiska displayen)
ADISP1:	15000 (Alfanumeriskdisplay)
RAM:	20000 (Ram-minnet)

Pal programmen är skrivna med utgång från ovanstående mapp.
De finns bifogade i appnedix.

Mjukvaran i övrigt har utvecklats i ANCI-C.
Vi bestämde oss för att utveckla mjukvaran så att den var interruptstyrd. Tiden medgav inte att göra en realtidskärna.

Då utvecklingen fortfarande pågår kan vi endast beskriva mjukvaran i form av pseudokod.

main metod:
Nollställer och ger startvärden till alla globala variabler.
Initierar både realtidsklockan, alfanumeriskdisplayen och den grafiska displayen.
Ritar ut den första skärmbilden.
Sätter CPU:n så den kan ta emot interrupts.
Sedan sätter vi igång en oänliglop.

Knapptrycknings interrupt (level 5)
Var gång vi trycker på en knapp skall denna tas hand om.
Beroende på vilken meny vi är i händer det olika saker.

A/D och time interrupt (level 2)
Vi tar hand om följande tidsinterrupt: sekund, minut, timme och dag.
Då vi får sekund-interrupt uppdaterar vi alla mätvärden.
Varannan sekund ritar vi ut kolonet mellan timmar och minuter i ”main menu”.
Var 10:e sekund ritar vi ut vindstyrkan (den medelvärdesbildas under 10 sek)
Var 30:e minut sparar vi undan mätvärden i en vektor. Denna vektor är 48 element stor så den täcker ett dygn.
Var dag sparar vi undan de medelvärdesbildade 30 minut värdena i en dagvektor med 365 värden. Denna vektor använd sedan för att representera ”average month” och ”average year” samt för att kunna hämta data under ”Get data”.
Vad gäller A/D-interrupt så får vi bara det då A/D-omvandlaren är klar med en bestämd omvandling, vi får en så kallad ”End Of Conversion”.

För att kunna rita bilder och grafik på vår grafiska display tog vi hjälp av vår kompis som heter Robert Alm. Han konstruerade ett matlab-program som tar en 64x128 stor svartvit BMP bild och kodar den i rätt format. Ett stort tack! Matlab-programmet är bifogat i appendix.

Resultat och tankar

Hårdvaruutvecklingen var ganska rätt fram. Man valde kretsar utifrån de specifikationer vi satte. Virningen var ganska enkel, vi startade med spänningsmatningen, efter det tog vi oss an adress- och databuss.
Sedan virade vi kontrollsignalerna. Då vi gick på problem konsulterade vi vår handledare. Efter två veckors arbete var vi ”färdiga” med konstruktionen.

Nästa steg var att skriva kod och bränna PAL:arna. Detta var ganska rätt fram det också.

Det sista var att skriva programmet. Här har vi dykt på det konstiga problemet att vår kompilator är konstig. Man kan ej testa lokala variabler i if-satser. Vilket leder till att man deklarerar många variabler globalt.
Annars anser vi att vi lyckats göra en snygg layout och ett användarvänligt gränssnitt.

Då man tar sig an ett sådant här projekt anser man dock snabbt att man aldrig blir färdig med utvecklingen. Men man lär sig ofantligt mycket.

Vi vill tacka vår handledare för fint samarbete samt Robban som hjälpt oss med grafiken och en del experthjälp vad gäller ANCI-C.

Appendix A

Kopplingsschema.

Appendix B

```
Title       PAL1

device      22V10

'input
AS          1           'AddressStrobe from CPU
A12         2
A13         3
A14         4
A15         5
A16         6
A17         7
DS          8           'DataStrobe from CPU
RW          9           'ReadWrite from CPU
DAPAL2      10          'DataTransferAcknowledge from PAL2

'11 och 13 Free for use

GND         12

'output
CSROM       14          'ChipSelect EPROM
CSRAM       15          'ChipSelect SRAM
CSRTC       16          'ChipSelect RealTimeClock
CSKEY       17          'ChipSelect KeyDecode
CSCNT       18          'ChipSelect 8-bit Counter
OE          19          'OutputEnable to everything ;)
WE          20          'WriteEnable to everything ;)
DTACK       21          'DataTransferAcknowledge to CPU
ADSTART     22          'StartConversion to A/D
OEAD        23          'OutputEnable to A/D

VCC         24


start
CSROM /= /AS * /A17 * /A16 * /A15 * /A14 * /A13 * /A12;
CSRAM /= /AS * A17;
CSRTC /= /AS * /A17 * /A16 * A15 * /A14 * /A13 * /A12;
CSKEY /= /AS * /A17 * /A16 * A15 * A14 * /A13 * /A12;
CSCNT /= /AS * /A17 * /A16 * A15 * A14 * A13 * A12;
OE /= /DS * RW;
WE /= /DS * /RW;
DTACK /=  /CSROM + /CSRAM + /CSRTC + /CSKEY + /CSCNT + ADSTART +
OEAD;
ADSTART = /AS * /A17 * /A16 * A15 * /A14 * A13 * /A12;
OEAD = /AS * /A17 * /A16 * A15 * /A14 * A13 * A12;

end
```

```
Title       PAL2

device      22V10

'input
AS          1           'AddressStrobe from CPU
A12         2
A13         3
A14         4
A15         5
A16         6
A17         7
FC0         8           'Processor status
FC1         9           'Processor status
FC2         10          'Processor status
CPUE        11          'E from Processor

GND         12

VPANET      13          'Asyncronus bus ---> Syncronus bus to Disp.



'output
CSDISP      14          'ChipSelect Dips.
CPUVPA      15          'VPA to CPU
CS1GD       16          'Chipselect 1 to GraficDisp.
CS2GD       17          'Chipselect 2 to GraficDisp.
EALFAD      18          'Enable to AlfaNumer. Disp.

'input
VMANET      23          'VMA from syncronus Net

VCC         24


start
CSDISP = /AS * /A17 * A16 * /A15 * /A14 * /A13 * A12
             + /AS * /A17 * A16 * /A15 * /A14 * A13 * A12
             + /AS * /A17 * A16 * /A15 * A14 * /A13 * A12;
CPUVPA /= /AS * FC0 * FC1 * FC2 + /VPANET;
CS1GD /= /AS * /A17 * A16 * /A15 * /A14 * /A13 * A12 * /VMANET;
CS2GD /= /AS * /A17 * A16 * /A15 * /A14 * A13 * A12 * /VMANET;
EALFAD = /AS * /A17 * A16 * /A15 * A14 * /A13 * A12 * CPUE * /VMANET;
end
```

```
Title       PAL3

device      22V10

'input
AS          1           'AddressStrobe from CPU
A12         2
A13         3
A14         4
A15         5
A16         6
A17         7
FC0         8           'Processor status
FC1         9           'Processor status
FC2         10          'Processor status
INTRTC      11          'Interrupt from RealTimeClock

GND         12

INTKEY      13          'Interrupt from KeyDecoder


'output
IPL02       14          'Interrupt 0/2 on CPU
IPL1        15          'Interrupt 1 on CPU

'input
INTAD       23          'Interrupt form A/D

VCC         24


start
IPL02 /= INTKEY;
IPL1 /= /INTRTC * /INTKEY + INTAD * /INTKEY;
end
```

Appendix C

```
function bildkonvert(filenamein, filenameleft, filenameright);
%Skapa bild i specialformat för LCD-display.
% Use: bildkonvert('infilename','leftfilename','rightfilename')

picture = double(imread(filenamein));

figure(1)
```

```
colormap gray;
imagesc(picture)

picture(find(picture==0)) = 2;
picture(find(picture==1)) = 0;
picture(find(picture==2)) = 1;

fout = fopen(filenameleft, 'w');

for rad=0:7
   for byte=1:64
      B = 0;%uchar(0);
      for bit=1:8
         B = bitset(B, bit, picture(((rad*8) + bit), byte));
      end
      fwrite(fout,B,'uchar');
   end
end

fclose(fout);
fout = fopen(filenameright, 'w');

for rad=0:7
   for byte=65:128
      B = 0;%uchar(0);
      for bit=1:8
         B = bitset(B, bit, picture(((rad*8) + bit), byte));
      end
      fwrite(fout,B,'uchar');
   end
end

fclose(fout);

%läs in och återskapa bild för att visa hur det ser ut
fin = fopen(filenameleft, 'r');
left = zeros(64,64);
for rad=0:7
   for byte=1:64
      B = double(fread(fin,1));
      for bit=1:8
         left(((rad*8) + bit), byte) = bitget(B, bit);
      end
   end
end

fclose(fin);
fin = fopen(filenameright, 'r');
right = zeros(64,64);
for rad=0:7
   for byte=1:64
      B = double(fread(fin,1));
      for bit=1:8
         right(((rad*8) + bit), byte) = bitget(B, bit);
      end
   end
```

```
end

fclose(fin);

left(find(left==0)) = 2;
left(find(left==1)) = 0;
left(find(left==2)) = 1;

right(find(right==0)) = 2;
right(find(right==1)) = 0;
right(find(right==2)) = 1;

figure(2);
colormap gray;
subplot(1,2,1);
imagesc(left);
subplot(1,2,2);
imagesc(right);
```

Appendix D

C-Code