Institution of Information Technology

Report of Digital Project

Supervisor: Bertil Lindvall

TIME RELATED WEATHER STATION

Project Group 10

Hongwu Tong e98

Ingemar Lind e98

Date: 2002-03-01

Abstract

This report describes the design process of constructing a time related weather station, including both hardware and software. A digital circuit has been designed to present the temperatures of indoors and outdoors as a function of the time. The circuit will generate an alarm signal if the temperature is beyond a predefined temperature range. It can calculate the mean temperature and the maximum or minimum temperature in a day or a month. The main part of the hardware is the processor, MC68008 with 48 pins, which communicates with the peripheral components. The programmable logic components, PALCE22V10, are used to generate chip's select-signals for different peripheral components and the interrupt-signals for the processor. The software is written in C and Assembler program language. The whole construction has been tested by a development system for MC68008, which can help the designer to debug the hardware and software by using different kind of commands.

Contents

Introduction

The project is an assignment of the course, called digital projects. The purpose of the course is to illustrate industrial development. The project aims towards designing a prototype for further development with necessary documentation. The main part of the course consists of designing, building and testing the construction.

The goal of the project is to construct a time related weather station by using a MC68008 processor. The specifications are as follows:

- Two temperatures, indoors and outdoors, should be measured every one minute and presented, together with the date and time.
- The user should be able to set values for max- and min-temperatures. If the temperature is beyond the temperature range, a light diode should be lit as an alarm signal.
- The station should be able to present the maximum and minimum temperature values on the present day, and in the present month, together with the date and time. It should also be able to present the mean temperature during the present day and the present month.

In the first chapter the hardware design is described, the software design is described in the second chapter, after that the problems are discussed and some conclusions are reached. At last there are some appendixes with the schema of the hardware construction and the program code.

Hardware construction

Components

The data sheets of the all components used in the project can been found in the information bank on the homepage of the course.

Processor: MC68008

The MC68008 is a member of the M68000 Family of advanced microprocessors. It allows the design of cost effective systems using 8-bit data buses while providing the benefits of a 32-bit microprocessor architecture. The MC68008 is available in a 48-pin dual-in-line package and a 52-pin quad plastic package. In this project a 48-pin dual-in-line package is used. The MC68008 works in an asynchronous mode in this project. It has a 20- bit address bus and an 8- bit data bus.

PAL: PALCE22V10

The designer should decide at which addresses the different peripheral components will be placed. Every component needs one or more unique addresses depending on the internal structure. In order to reach the goal, the programmable logic PALCE22V10 is used to generate the signals that the peripheral components need. It is used extensively because of its advantages in making the constructing easier.
The PALCE22V10 has 22 pins inputs and 10 outputs. Two PALCE22V10 are used in the design. The first one decodes the addresses and sets the control signals to the peripheral units, i.e. EPROM, SRAM, Real-Time Clock, A/D Converter, LATCHES and sends the DTACK signal (active low I/O access) to the processor. The second PAL sends the control-signals to the display and 16-Key Encoder and generates the interrupt-signals that are from the Real-Time Clock (level 5) and the keyboard (level 2). The PAL programs can be found in Appendix B.

Program Memory: EPROM 27C128

The EPROM 27C128 is a high performance128K UV Erasable Electrically Programmable Read Only Memory. The program code, which controls the construction, has been saved in the memory.

Memory: SRAM 6264

The Hitachi HM6264BI is 64k-bit static RAM organized 8-kword × 8-bit. It is used to save all measure values, stack and program variables during the execution. It is both readable and writeable.

Real-Time Clock: ICM7170

The ICM7170 real time clock is a microprocessor bus compatible peripheral, fabricated using Harris' silicon gate CMOS LSl process. An 8-bit bi-directional bus is used for the data I/O circuitry. The clock is set or read by accessing the 8 internal separately addressable and programmable counters from 1/100 seconds to years. An extremely stable oscillator frequency is achieved through the use of a one-chip regulated power supply. It generates periodic interrupt, which is used to update the temperature measurement.

A/D Converter: ADC0804

The ADC0804 is an 8-bit successive approximation A/D converter. It converts the analog signal from the temperature sensor to the digital signal. It works in free-running mode in order to get the measure values quickly.

Temperature Sensor: LM335

In order to measure the temperature below the zero, the LM335 calibrated in °Kelvin temperature sensor is used in the project. It has a breakdown voltage directly proportional to the absolute temperature at 10 mV/K.

Display: Dot-Matrix LCD units

The LCD unit receives character codes (8 bits per character) from a microprocessor or microcomputer, latches the codes to its display data RAM (80-byte DD RAM for storing 80 characters), transforms each character code into a 5 × 7 dot-matrix character pattern, and displays the characters on its LCD screen.

16-Key Encoder: MM74C922

This CMOS key encoder provides all the necessary logic to fully encode an array of SPST switches. A Data Available output goes to a high level when a valid keyboard entry has been made. The Data Available output returns to a low level when the entered key is released.

D-TYPE FLIP-FLOP: SN74HC74

The SN74HC74 contains two independent D-type positive-edge-triggered flip-flops. Only one D- flip-flop is used in the project in order to generate a short pulse signal to the PAL2, which generates further keyboard interrupt signal to the processor.

D-TYPE LATCH: SN74HC573A

While the latch-enable (LE) input is high, the Q outputs respond to the data (D) inputs. When LE is low, the outputs are latched to retain the data that was set up. If the temperature is beyond the predefined temperature range, the processor will generate a signal, which goes through the D-latch to light the diode.

Schema construction

The whole hardware construction is drawn in the program PAD:s Power Logic and the schema can be seen in Appendix A. The schema has been improved during the process of the construction. From the schema, one can see that all units communicate via an 8-bit data bus. In order to connect all units' data outputs to the data bus, it requires that data outputs have "tristate" level. It means that chips with this functionality cannot affect the signals in the bus until they become active. The address bits A12-A0 present the 8192 different bytes within the memory area. A13-A16 is used to decode the memory area for different chips.
When e.g. the EPROM is accessed, the address bus set the address to this unit. The PAL decodes that address and activates the control signals to the EPROM and at the same time deactivates the control signals to all the other peripheral units.

Address map

The following figure shows the memory map.

0	4000	6000	8000	A000	C000	E000	10000
EPROM	A/D 1 (indoors)	A/D 2 (outdoors)	Real-time clock	Display	Latch	Encoder	SRAM

Building and testing the hardware construction

The hardware has been built on a board. The components are placed in sockets in a good ordering and have mainly been connected to each other by a winding technique. A thin cable is winded around the pin of the socket with a tool. Some parts, mainly the ground and the power pins have been soldered on the board. The address bus and data bus have been winded first and then the control signals in order to easily change the connection by debugging, because mostly some control signals should be changed.
After building the hardware, it has been tested with a development system called it68, which can help the designer to debug the hardware by using different kind of commands. The components have been tested one by one until all components work well.

Software construction

When we had finally managed to get the hardware to work, we started with the software development. The main programming language used for this was C. Some instructions however, for example initialising interrupts and setting the stack pointer, had to be written in assembler.

Before starting to write the real program, we made a simple sketch of the program flow:

main method:

```
// set start values to all parameters.

// write initialization instructions to the display and the real-time clock.

// enable interrupts.

while(true) {  // unlimited loop: wait for interrupts }
```

timer interrupt method (interrupt level 5):

```
// read the current temperature outdoors and indoors from the A/D – converters.
// show the temperatures on the display together with the present date and time, if
// the program is currently in ”normal mode”.

// update the mean temperature during the present day and month.

// if the temperatures are higher/lower than the highest/lowest temperature of the
// present day or month: save the temperature together with the current time (if
// highest/lowest of the day) or the current date (if highest/lowest of the month).

// if the temperature is outside one of the temperature borders set by the user: light
// a warning lamp.
```

button interrupt method (interrupt level 2):

```
// call the button encoder to find out which button the user pressed.

// if (the button was not <ENTER>)
//         save the button in a vector.
// else
```

```
//      look in the vector at the buttons recently pressed to find out which
//      function the user desires, the entered value of the temperature border or
//      the date and time.

//      perform the desired function and update parameters.

//      reset the button vector.
```

Since the two interrupt methods were the biggest parts of the program, we decided to write one each.
While writing these methods, we found out that we could also need some more methods, which we also wrote. One important method was the wait() method.
When we send instructions to the real-time clock or the display, we have to wait for a small amount of time between two instructions, in order to let the component finish handling the previous one. In order to do this, we call the wait() method, which simply goes through some empty for-loops.
Some other methods we added were methods for writing an integer or a float to the display, methods for writing indoor and outdoor temperatures to the display together with date and time and methods for decoding the value of the stored buttons in the vector, depending on whether they were describing a choice, a temperature or date and time.

An important variable in the program is the functionSelect variable. This variable shows which function that has been selected, and thereby also in which state the program currently is. The three states are the following:

- "Normal mode": The program is showing the current temperature, date and time on the display. When the program is in this mode, functionSelect is equal to 0.

- "Old data mode": The program is showing minimum, maximum or mean temperature of the present day or month, or the selected maximum- or minimum temperature borders indoors and outdoors. When the program is in this mode, functionSelect is between 1 and 8.

- "Printing mode": The user is currently entering a new maximum- or minimum temperature border value or setting the time and date. When the program is in this mode, functionSelect is between 9 and 13.

Results

After about four weeks work, we have finished the whole hardware design, from writing the design specification, selecting components, drawing circuit schema, building the circuit on the board, connecting the different components together, testing the hardware and improving it. By debugging with the help of the development system, we found out that the oscillator didn't work. The reason is that the circuit is connected by a winding technique. The cable is too long and the components are not soldered on the board, so there is noise in the circuit. We changed it to an oscillator chip, which works well. The second problem is that the free running mode connection for the two A/Ds should be separately connected to the RESET in order to avoid interaction between the two A/Ds. After finishing the hardware design, we began to design the software, which is also an important part in the construction.

The software development took much less time than the hardware development. Within a week, we had a program that worked nicely in the testing environment.
However, when we tried to run the program from an EPROM, we ran into several problems. The first was that our program turned out to be bigger than we had expected and did not fit on the 8x8k EPROM. Therefore, we had to switch to a 16x8k EPROM. This caused some redesign of the hardware and reprogramming of one of our PAL units, since the new EPROM took twice as much address space as the old one.
However, the new version soon also worked in the test environment, but, for some reason, it did not work properly on the EPROM.

Improvements

Since we got our program to work fine in the test environment, we were not too disappointed. And since the program worked there, it was not altogether bad…
However, there were some things that could have been done better:

All the work in our program was carried out in the interrupt routines. Since interrupts should be handled as fast as possible, this was not the best solution. It would have been better to set a flag when an interrupt occurred, and then check the flag in the unlimited loop in the main program, in order to find out what should be done.
In our project, however, interrupts occurred rather seldom, so we never had any problems with this. But if we had written a program where many different kinds of interrupts occurred very often, we might have run into problems when using this method.

Another thing is that we should have written more comments in the program code, while writing it, in order to make it easier to read. Especially the places in the program, where we write a lot of instructions to the display and the real-time clock, are pretty hard to understand afterwards.

Conclusions

During the seven weeks we have learnt how one can design a digital circuit, practiced what we have learnt in previous courses, especially Design of Digital Circuits (Digitalteknik) and Computer Organization (Datorteknik), and improved our knowledge a lot. The Digital Project course is perhaps the most interesting course we have taken during our studying period in LTH.

References

[1] Data sheets in the information bank:
http://www.it.lth.se/it/courses/Digp/datablad/datablad_index.htmo

[2] It68 utvecklingssystem för MC68008
http://www.it.lth.se/digproj/PDF_files/it68.pdf

[3] MC68008 8-/32-BIT MICROPROCESSOR WITH 8-BIT DATA BUS

[4] Brian W. Kernigham / Dennis M. Ritchie: The C Programming Language
Second Edition, Prentice Hall Software Series

Appendix A: Circuit schema

Appendix B: Programmable logic

```
'PAL 1

device 22v10

CLK     1
A13     2
A14     3
A15     4
A16     5
AS      6               'aktivt låg
DS      7               'aktivt låg
RW      8               '0: skriv 1: läs
DTACK2          9       'aktivt låg
GND             12
CSPROM          14      'aktivt låg
CSRAM           15      'aktivt låg
CSTIMER         16      'aktivt låg
CSAD1           17      'aktivt låg
CSAD2           18      'aktivt låg
RD              19      'aktivt låg
WR              20      'aktivt låg
DTACK           21      'aktivt låg
CSLATCH         22      'aktivt hög
VCC             24

start

CSPROM /= /A14 * /A15 * /A16 * /AS;
CSAD1 /= /A13 * A14 * /A15 * /A16 * /AS;
CSAD2 /= A13 * A14 * /A15 * /A16 * /AS;
CSTIMER /= /A13 * /A14 * A15 * /A16 * /AS;
CSLATCH = /A13 * A14 * A15 * /A16 * /AS;
CSRAM /= /A13 * /A14 * /A15 * A16 * /AS;
RD /= RW * /DS;
WR /= /RW * /DS;
DTACK /= /CSPROM + /CSRAM + /CSAD1 + /CSAD2 + CSLATCH + /CSTIMER +
/DTACK2;

end
```

```
'PAL 2

device 22v10

CLK         1
A13         2
A14         3
A15         4
A16         5
FC0         6
FC1         7
FC2         8
AS          9       'aktivt låg
INTKNAPP    10
INTTIME     11      'aktivt låg
GND         12
DS          13
VPA         14      'aktivt låg
IPL02       15      'aktivt låg
IPL1        16      'aktivt låg
EDISP       17      'aktivt hög
OEKNAPP     18      'aktivt låg
DTACK2      19      'aktivt låg
VCC         24

start

VPA /= FC0 * FC1 * FC2 * /AS;
IPL1 /= INTKNAPP * INTTIME;
IPL02 /= /INTTIME;
EDISP = A13 * /A14 * A15 * /A16 * /DS;
OEKNAPP /= A13 * A14 * A15 * /A16 * /AS;
DTACK2 /= EDISP + /OEKNAPP;

end
```

Appendix C: C-code

```
/*************************************************************************
                    test.c
                    ------
  Detta ar huvudprogrammet som ar skrivet i ANSI C. Exekveringen av hela
  programpaketet borjar i pmain.68k (lage __main).

  exp4() anropas fran assemblyprogrammet exp4.68k vid avbrott.

  _avben() anropar avben.68k vilket tillater avbrott fran PI/T.
*************************************************************************/
#include <math.h>

int tid;

float maxtempin, mintempin, maxtempout, mintempout;

unsigned short int *adin, *adout, *display, *encoder, *latch, *timer;
float intemp,outtemp;
float maxdi;
float mindi;
float maxdu;
float mindu;
float maxmi;
float minmi;
float maxmu;
float minmu;
int maxMonthi,maxMonthu,minMonthi,minMonthu, maxDayu,minDayu,maxDayi,minDayi;


float meanTempIn;
float meanTempOut;
float meanTempInMon;
float meanTempOutMon;
int maxhouri, maxminutei, minhouri, minminutei,maxhouru, maxminuteu, minhouru,
minminuteu,functionSelect;
int second, hour, minute, day, month;
int nbrOfTemps;
int nbrOfDays;

int buttons[10];            /* initiera alla element till -1 */
int nrButtons;              /* antal knapptryckningar sedan senaste <ENTER>*/


void wait(int time);
void printTemp(int temp);
void printFloatTemp(float temp, int precision);
void dispMaxMinTemp (float maxmindi,float maxmindu, int maxminhouri, int maxminminutei, int
maxminhouru, int maxminminuteu);
void dispMeanValue (float meanIn, float meanOut);
void normalSituation (void);
void decodeDate(void);
float decodeTemp(void);
int decodeChoice(void);
```

```
void performFunction(void);
void writeButtonToDisplay(int button);
void init(void);


main()
{

        tid = 50;

        functionSelect=0;
        nrButtons = 0;

        nbrOfTemps=0;
        nbrOfDays=0;

        meanTempIn = 0.0;
        meanTempOut = 0.0;
        meanTempInMon = 0.0;
        meanTempOutMon = 0.0;

        maxtempin = 100.0;
        mintempin = -100.0;
        maxtempout = 100.0;
        mintempout = -100.0;

        maxdi =0.0;
        mindi = 100.0;
        maxdu =0.0;
        mindu = 100.0;
        maxmi =0.0;
        minmi = 100.0;
        maxmu =0.0;
        minmu = 100.0;

        adin = (unsigned short int *) 0x4000;          /* AD 1*/
        adout = (unsigned short int *) 0x6000;         /* AD 2*/
        timer = (unsigned short int *) 0x8000;
        display = (unsigned short int *) 0xa000;
        latch = (unsigned short int *) 0xc000;
        encoder = (unsigned short int *) 0xe000;


        *display = 0x38;
        wait(tid);
        *display = 0x0e;
        wait(tid);
        *display = 0x01;
        wait(tid);

        *(timer + 0x11) = 0x1c;
        wait(tid);
        *(timer + 0x10) = 0x10;
        wait(tid);
```

```
        init();
        wait(tid);

        _avben();           /* tillat avbrott */



        while(1) /* evig loop */
        {

        }

}


void exp2(void)             /* avbrottsprogram knappavbrott*/
{
        int i, fs;
        float temp;
        unsigned short int thebutton = *encoder % 16;
        wait(tid);

        if(thebutton < 15)      /* 15: <ENTER> */
        {
                if(nrButtons < 9)   /* annars: felaktig inmatning */
                {
                buttons[nrButtons] = thebutton;   /* spara knapptryckningen */
                nrButtons++;
                        if(functionSelect >= 9)
                                writeButtonToDisplay(thebutton);
                }
        }
        else                            /* inmatning klar: undersök vad som skrivits in */
        {
                if(nrButtons < 9 && nrButtons > 0)   /* annars: felaktig inmatning */
                {
                        if(functionSelect <= 8)      /* inmatning av funktion */
                        {
                                fs = decodeChoice();
                                if(fs >= 0 && fs <= 13)
                                        functionSelect = fs;
                        }
                        else if(functionSelect >= 9 && functionSelect <= 12)  /* inmatning av larmgräns */
                        {
                                temp = decodeTemp();
                                if(temp > -300.0)
                                {
                                        if(functionSelect == 9)
                                                maxtempin = temp;
                                        else if(functionSelect == 10)
                                                mintempin = temp;
                                        else if(functionSelect == 11)
                                                maxtempout = temp;
                                        else if(functionSelect == 12)
                                                mintempout = temp;
                                }
```

```
                                        functionSelect = 0;
                                }
                                else
                                {
                                        decodeDate();
                                        functionSelect = 0;
                                }
                        }
                        else
                                functionSelect = 0;

                        for(i = 0; i < 10; i++)
                                buttons[i] = -1;
                        nrButtons = 0;

                        performFunction();          /* utför olika operationer beroende på vad funcSelect är */
                }
}

void exp5(void)             /* avbrottsprogram timeravbrott*/
{
                        init();
}

void wait(int time)
{
        int i,j;
        for(i = 0; i < time; i++)
        {
                for(j = 0; j < 100; j++)
                        ;
        }
}

void printFloatTemp(float temp, int precision)
{
        float t2,t3;
        int t4,t5;
        int i;
        int t1 = (int)temp;
        t2 = (temp - (float)t1);



        if(t2 < 0.0)
                t2 = t2*(-1.0);

        for(i = 0; i < precision; i++)
                t2 = t2*10.0;


        t5 =(int)t2;

        t3 =(t2 - (float)t5);

        if (t3>=0.5)
```

```
	{
		t4 = t5 +1;

		if(t4 == pow(10, precision))
		{
			t4 = 0;
			t1 = t1 + 1;
		}

	}
	else
		t4=t5;

	printTemp(t1);
	wait(tid);
	*(display+1) = 0x2e;

	printTemp(t4);

}

void printTemp(int temp)
{

	int t, t2;

	if(temp < 0)
	{
		wait(tid);
		*(display+1) = 0x2d;
		temp = temp*(-1);
	}

	t = temp/10;
	t2 = temp%10;

	if(t > 0)
		printTemp(t);
	wait(tid);
	*(display+1) = (0x30 + t2);
}




void dispMaxMinTemp (float maxmindi,float maxmindu, int maxminhouri, int maxminminutei, int
maxminhouru, int maxminminuteu)
{
	wait(tid);
	*display = 0x01;
	wait(tid);
	*(display+1) = 0x49;
	wait(tid);
	*(display+1) = 0x3a;
	wait(tid);
```

```
        printFloatTemp(maxmindi, 1);
        wait(tid);
        *(display+1) = 0xdf;
        wait(tid);
        *(display+1) = 0x43;
        wait(tid);
         *(display+1) = 0xa0;
        wait(tid);


        printTemp(maxminhouri);
        wait(tid);
        if (functionSelect == 1 || functionSelect == 2)
        {
                *(display+1) = 0x3a;
                wait(tid);
                if (maxminminutei < 10)
                        *(display+1) = 0x30;
        }
        else
                *(display+1) = 0x2f;
        wait(tid);
        printTemp(maxminminutei);
        wait(tid);
        *(display) = 0xc0;
        wait(tid);
        *(display+1) = 0x55;
        wait(tid);
        *(display+1) = 0x3a;
        wait(tid);
        printFloatTemp(maxmindu, 1);
        wait(tid);
        *(display+1) = 0xdf;
        wait(tid);
        *(display+1) = 0x43;
        wait(tid);
        *(display+1) = 0xa0;
        wait(tid);

        printTemp(maxminhouru);
        wait(tid);
        if (functionSelect == 1 || functionSelect == 2)
        {
                *(display+1) = 0x3a;
                wait(tid);
                if (maxminminuteu < 10)
                        *(display+1) = 0x30;
        }
        else
                *(display+1) = 0x2f;
        wait(tid);
        printTemp(maxminminuteu);
}

void dispMeanValue (float meanIn, float meanOut)
```

```
{
	wait(tid);
	*display = 0x01;
	wait(tid);
	*(display+1) = 0x49;
	wait(tid);
	*(display+1) = 0x3a;
	wait(tid);
	printFloatTemp(meanIn, 1);
	wait(tid);
	*(display+1) = 0xdf;
	wait(tid);
	*(display+1) = 0x43;
	wait(tid);
	*(display) = 0xc0;
	wait(tid);
	*(display+1) = 0x55;
	wait(tid);
	*(display+1) = 0x3a;
	wait(tid);
	printFloatTemp(meanOut, 1);
	wait(tid);
	*(display+1) = 0xdf;
	wait(tid);
	*(display+1) = 0x43;
}


void normalSituation(void)
{
		wait(tid);
		*display = 0x01;
		wait(tid);
		*(display+1) = 0x49;
		wait(tid);
		*(display+1) = 0x3a;
		wait(tid);
		printFloatTemp(intemp, 1);
		wait(tid);
		*(display+1) = 0xdf;
		wait(tid);
		*(display+1) = 0x43;
		wait(tid);
		*(display+1) = 0xa0;
		wait(tid);
		printTemp(hour);
		wait(tid);
		*(display+1) = 0x3a;
		wait(tid);
		if (minute <10)
		{
			*(display+1) = 0x30;
			wait(tid);
		}
		printTemp(minute);
		wait(tid);
```

```
		*(display) = 0xc0;
		wait(tid);
		*(display+1) = 0x55;
		wait(tid);
		*(display+1) = 0x3a;
		wait(tid);
		printFloatTemp(outtemp, 1);
		wait(tid);
		*(display+1) = 0xdf;
		wait(tid);
		*(display+1) = 0x43;
		wait(tid);
		*(display+1) = 0xa0;
		wait(tid);
		printTemp(month);
		wait(tid);
		*(display+1) = 0x2f;
		wait(tid);
		printTemp(day);
}



void decodeDate(void)
{
	int i, newmonth, newdate, newhour, newminute;

	for(i = 0; i < nrButtons; i++)
	{
		if(buttons[i] > 9)
		{
			return;
		}
	}
	if(nrButtons == 8)
	{
		newmonth = 10*buttons[0] + buttons[1];
		if(newmonth > 12)
		{
			return;
		}

		newdate = 10*buttons[2] + buttons[3];
		if(newdate > 31)
		{
			return;
		}

		newhour = 10*buttons[4] + buttons[5];
		if(newhour > 23)
		{
			return;
		}
		newminute = 10*buttons[6] + buttons[7];
		if(newminute > 59)
```

```
                {
                        return;
                }

                        /* tidsställning samt nollställning av mätvärden... */

                wait(tid);
                *(timer +0x04) = newmonth;
                wait(tid);
                *(timer +0x05) = newdate;
                wait(tid);
                *(timer +0x01) = newhour;
                wait(tid);
                *(timer +0x02) = newminute;
                wait(tid);

                month = newmonth;
                day = newdate;
                hour = newhour;
                minute = newminute;

                nbrOfTemps = 0;
                nbrOfDays = 0;
                maxdi = 0.0;
                mindi = 100.0;
                maxdu = 0.0;
                mindu = 100.0;
                maxmi = 0.0;
                minmi = 100.0;
                maxmu = 0.0;
                minmu = 100.0;
                meanTempIn = 0.0;
                meanTempOut = 0.0;
                meanTempInMon = 0.0;
                meanTempOutMon = 0.0;


        }
}

int decodeChoice(void)
{
        if(nrButtons == 0 || nrButtons > 2 || buttons[0] > 9 || buttons[1] > 3)
                return -1;
        else
        {
                if(nrButtons == 2)
                        return(10*buttons[0] + buttons[1]);
                else
                        return buttons[0];
        }
}

float decodeTemp(void)
{
        int i;
```

```
	int komma = nrButtons, sign = 1;
	float temp = 0.0;

	if(buttons[0] == 12 || (nrButtons == 1 && buttons[0] > 9) || (buttons[0] > 9 && buttons[1] > 9))
		return -300.0;

	for(i = 1; i < nrButtons; i++)
	{
		if(buttons[i] == 12)		/* komma */
		{
			if(komma == nrButtons)
				komma = i;
			else
				return -300.0;
		}
		else if(buttons[i] > 9)
			return -300.0;

	}

	if(buttons[0] == 11) /* minus */
		sign = -1;


	for(i = 0; i < nrButtons; i++)
	{
		if(buttons[i] <= 9)
		{
			if(i < komma)
				temp += pow(10, (komma - 1 - i)) * (float) buttons[i];
			else
				temp += pow(10, (komma - i)) * (float) buttons[i];
		}
	}
	temp = temp * sign;

	return temp;
}


void writeButtonToDisplay(int button)
{
	if(button <= 9)
		*(display + 1) = (0x30 + button);
	else if(button == 10)					/* plus */
		*(display + 1) = 0x2b;
	else if(button == 11)					/* minus */
		*(display + 1) = 0x2d;
	else if(button == 12)					/* komma */
		*(display + 1) = 0x2e;
}

void performFunction(void)
{

	if(functionSelect == 0)
```

```
                normalSituation();
        else if(functionSelect == 1)
                dispMaxMinTemp(maxdi, maxdu, maxhouri, maxminutei, maxhouru, maxminuteu);
        else if(functionSelect == 2)
                dispMaxMinTemp(mindi, mindu, minhouri, minminutei, minhouru, minminuteu);
        else if(functionSelect == 3)
                dispMeanValue(meanTempIn, meanTempOut);
        else if(functionSelect == 4)
                dispMaxMinTemp(maxmi, maxmu, maxMonthi, maxDayi, maxMonthu, maxDayu);
        else if(functionSelect == 5)
                dispMaxMinTemp(minmi, minmu, minMonthi, minDayi, minMonthu, minDayu);
        else if(functionSelect == 6)
                dispMeanValue(meanTempInMon, meanTempOutMon);
        else if(functionSelect == 7)
                dispMeanValue(maxtempin, maxtempout);
        else if(functionSelect == 8)
                dispMeanValue(mintempin, mintempout);
        else if(functionSelect >= 9 && functionSelect <=13)
        {
                wait(tid);
                *display = 0x01;
                wait(tid);
                *(display + 1) = 0x3e;
        }

}

void init(void)
{
                int a;

                int light=0;

                wait(tid);
                a = *(timer + 0x10);
                wait(tid);
                intemp = *adin;
                intemp = intemp -123.15;
                wait(tid);
                outtemp = *adout;
                outtemp = outtemp - 123.15;
                wait(tid);
                second=*timer;
                wait(tid);
                hour=*(timer+01);
                wait(tid);
                minute=*(timer+02);
                wait(tid);
                month=*(timer+0x04);
                wait(tid);
                day=*(timer+0x05);
                wait(tid);

                if (functionSelect == 0)
                {
                        wait(tid);
```

```
		*display = 0x01;
		wait(tid);
		*(display+1) = 0x49;
		wait(tid);
		*(display+1) = 0x3a;
		wait(tid);
		printFloatTemp(intemp, 1);
		wait(tid);
		*(display+1) = 0xdf;
		wait(tid);
		*(display+1) = 0x43;
		wait(tid);
		*(display+1) = 0xa0;
		wait(tid);
		printTemp(hour);
		wait(tid);
		*(display+1) = 0x3a;
		wait(tid);
		if (minute <10)
		{
			*(display+1) = 0x30;
			wait(tid);
		}
		printTemp(minute);
		wait(tid);
		*(display) = 0xc0;
		wait(tid);
		*(display+1) = 0x55;
		wait(tid);
		*(display+1) = 0x3a;
		wait(tid);
		printFloatTemp(outtemp, 1);
		wait(tid);
		*(display+1) = 0xdf;
		wait(tid);
		*(display+1) = 0x43;
		wait(tid);
		*(display+1) = 0xa0;
		wait(tid);
		printTemp(day);
		wait(tid);
		*(display+1) = 0x2f;
		wait(tid);
		printTemp(month);
}

wait(tid);
if(intemp > maxtempin || intemp < mintempin)
		light = light + 1;

if(outtemp > maxtempout || outtemp < mintempout)
		light = light + 2;

*latch = light;
wait(tid);
```

```
		if (intemp > maxdi)
		{
			maxdi = intemp;
			maxhouri = hour;
			maxminutei = minute;
		}
		if (intemp < mindi)
		{
			mindi = intemp;
			minhouri = hour;
			minminutei = minute;
		}
		if (outtemp > maxdu)
		{
			maxdu = outtemp;
			maxhouru = hour;
			maxminuteu = minute;
		}
		if (outtemp < mindu)
		{
			mindu = outtemp;
			minhouru = hour;
			minminuteu = minute;
		}
		if (intemp > maxmi)
		{
			maxmi = intemp;
			maxMonthi = month;
			maxDayi = day;
		}
		if (intemp < minmi)
		{
			minmi = intemp;
			minMonthi = month;
			minDayi = day;
		}
		if (outtemp > maxmu)
		{
			maxmu = outtemp;
			maxMonthu = month;
			maxDayu = day;
		}
		if (outtemp < minmu)
		{
			minmu = outtemp;
			minMonthu = month;
			minDayu = day;
		}

		meanTempIn = meanTempIn / (nbrOfTemps +1) * nbrOfTemps + intemp / (nbrOfTemps
+1);
		meanTempOut = meanTempOut / (nbrOfTemps +1) * nbrOfTemps + outtemp /
(nbrOfTemps +1);
```

```
            meanTempInMon = meanTempInMon / (nbrOfDays +1) * nbrOfDays + meanTempIn  /
(nbrOfDays +1);
                meanTempOutMon = meanTempOutMon / (nbrOfDays +1) * nbrOfDays +
meanTempOut  / (nbrOfDays +1);

                if (hour == 0 && minute ==0)        /* ny dag */
                {
                        nbrOfTemps = 1;
                        nbrOfDays ++;
                        meanTempIn = intemp;
                        meanTempOut = outtemp;
                        maxdi = intemp;
                        mindi = intemp;
                        maxdu = outtemp;
                        mindu = outtemp;
                        maxhouri = hour;
                        maxminutei = minute;
                        minhouri = hour;
                        minminutei = minute;
                        maxhouru = hour;
                        maxminuteu = minute;
                        minhouru = hour;
                        minminuteu = minute;


                        if (day == 1)                   /* ny månad */
                        {
                                nbrOfDays = 1;
                                meanTempInMon = intemp;
                                meanTempOutMon = outtemp;
                                maxmi = intemp;
                                minmi = intemp;
                                maxmu = outtemp;
                                minmu = outtemp;
                                maxMonthi = month;
                                maxDayi = day;
                                minMonthi = month;
                                minDayi = day;
                                maxMonthu = month;
                                maxDayu = day;
                                minMonthu = month;
                                minDayu = day;
                        }


                }
                else
                        nbrOfTemps ++;

                wait(10*tid);
}
```

Appendix D: Simple user's manual

Enter your choice on the keyboard, followed by <ENTER>

0: Set the program in normal mode, i.e. show the present temperature indoors and outdoors, together with the current date and time.
When started, the program is in this mode.

1: Show the maximum temperature values of the present day indoors and outdoors, together with the actual time.

2: Show the minimum temperature values of the present day indoors and outdoors, together with the actual time.

3: Show the mean temperature values of the present day indoors and outdoors.

4: Show the maximum temperature values of the present month indoors and outdoors, together with the actual date.

5: Show the minimum temperature values of the present month indoors and outdoors, together with the actual date.

6: Show the mean temperature values of the present month indoors and outdoors.

7: Show the maximum temperature borders indoors and outdoors.

8: Show the minimum temperature borders indoors and outdoors.

9: Enter a new maximum temperature border indoors.

10: Enter a new minimum temperature border indoors.

11: Enter a new maximum temperature border outdoors.

12: Enter a new minimum temperature border outdoors.

13: Set the date and time. The input format is <mon-mon-day-day-hour-hour-min-min> followed by <ENTER>.