# PROTOTYPE TEMPERATURE CONTROL SYSTEM FOR COMBINED HYPERTHERMIA / PDT TREATMENT

Diploma paper

by
Niklas Hildebrand

Lund Reports on Atomic Physics, LRAP-115
Lund , September 1990

# PROTOTYPE TEMPERATURE CONTROL SYSTEM FOR COMBINED HYPERTHERMIA/PDT TREATMENT

CONTENTS PAGE



**DEV NOTE:** Refers to information meant for a future developer.

# 1 INTRODUCTION

## 1.1 The purpose of this work

Photodynamic therapy (PDT) has over the last few years proved to be an effective and interesting way of treating superficial malignant tumours. When treating thick tumours, however, the PDT laser is unable to penetrate the tissue enough to kill the deepest lying cells. In order to reach a higher efficiency, a combined treatment using both PDT and hyperthermia might be the answer. This work contains a description of a computerized temperature control system that can be used in connection with such a treatment.

## 1.2 Photodynamic therapy ( PDT )

During photodynamic treatment the drug hematoporphyrin derivative (HpD) is administered to the patient and distributed evenly throughout the body by the vascular system. After 2-3 days the drug has been washed out again apart from tumours ( due to various circumstances such as, for example, a chemical reorganization resulting from differences in pH values and lipophilicity [6]). HpD is a photosensitizing drug, i.e. it absorbs light of a particular wavelength (630 nm) and transfers that energy to the oxygen in the tissue. The oxygen changes from the normal state to a highly aggressive one that effectively destroys living tissue. A laser is ideal for this purpose as it is capable of producing light with high intensity within a narrow spectral range. The patient is exposed for 30-40 minutes of a power density of 10 - 50 $mW/cm^2$. After a few days cell necrosis can be seen [3].

## 1.3 Hyperthermia

Hyperthermia is a much older form of cancer treatment than photodynamic therapy. It has roots back in the 19:th century when it was observed that patients with malignant melanoma and inoperable sarcoma recovered from these tumours after periods of high fever [2]. A more controlled use of hyperthermia began at the turn of the century.

Local hyperthermia treatment uses a heat source to raise the temperature of the tumour to between 41 °C and 45 °C. Above a critical limit, about 43 °C, a majority of the exposed cells are killed and the recovery rate for cancer cells is less than for normal tissue [2]. The duration of the treatment is 1 - 2 hours. Fig 1 below shows the general principle of hyperthermia treatment.

There are many ways to accomplish hyperthermia but the clinically used methods are:

* Ultrasound ( frequencies 0.3 - 3 MHz )
* Electromagnetic fields ( radio frequencies below 300 MHz )
* Electromagnetic radiation ( microwave frequencies 300 - 2450 MHz )

It is also possible to use an IR source like a Nd-YAG laser or an ordinary lamp.

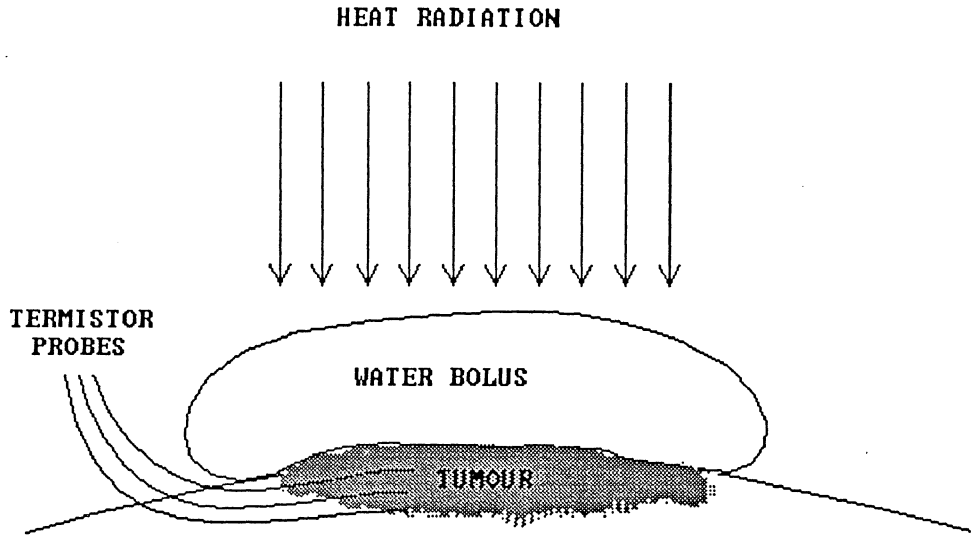



FIG 1. General principle of hyperthermia treatment.

Before the hyperthermia treatment begins, small electronic thermometers, called termistors, are inserted in the tumour so that the temperatures at various depths are known at all times. By letting a computer read the termistors and control the heat source and/or the surface cooling, an automatic hyperthermia system can be constructed.

Cancer cells can develop thermotolerance, i.e. they can stand heat better, if their temperature is raised considerably for a short time and then brought back to normal again. This thermotolerance ability makes it necessary to keep a stable temperature during the treatment [2].

1.4 Light penetration

The main mechanisms in the interaction of light with biological tissue is reflection, scattering and absorption. Which mechanism has the greatest influence is dependent on the wavelength of the light and on the molecular composition of the tissue [6]. Tissue contain various degrees of water, fat, proteins, haemoglobin and melanin depending on its function. Fig 2 below shows the molar extinction coefficient for different tissues and hematoporphyrin derivative at various wavelengths.

In the case of a combined treatment there is both the laser and the heat source to consider. The laser ( 630 nm ) has a penetration depth ( i.e. the depth where the intensity has fallen by a factor e ) of approximately 3 mm in basal cell carcinoma. 3-GHz microwaves has a penetration depth of 1 cm in muscle-like tissue and 8 cm in fatty [2].

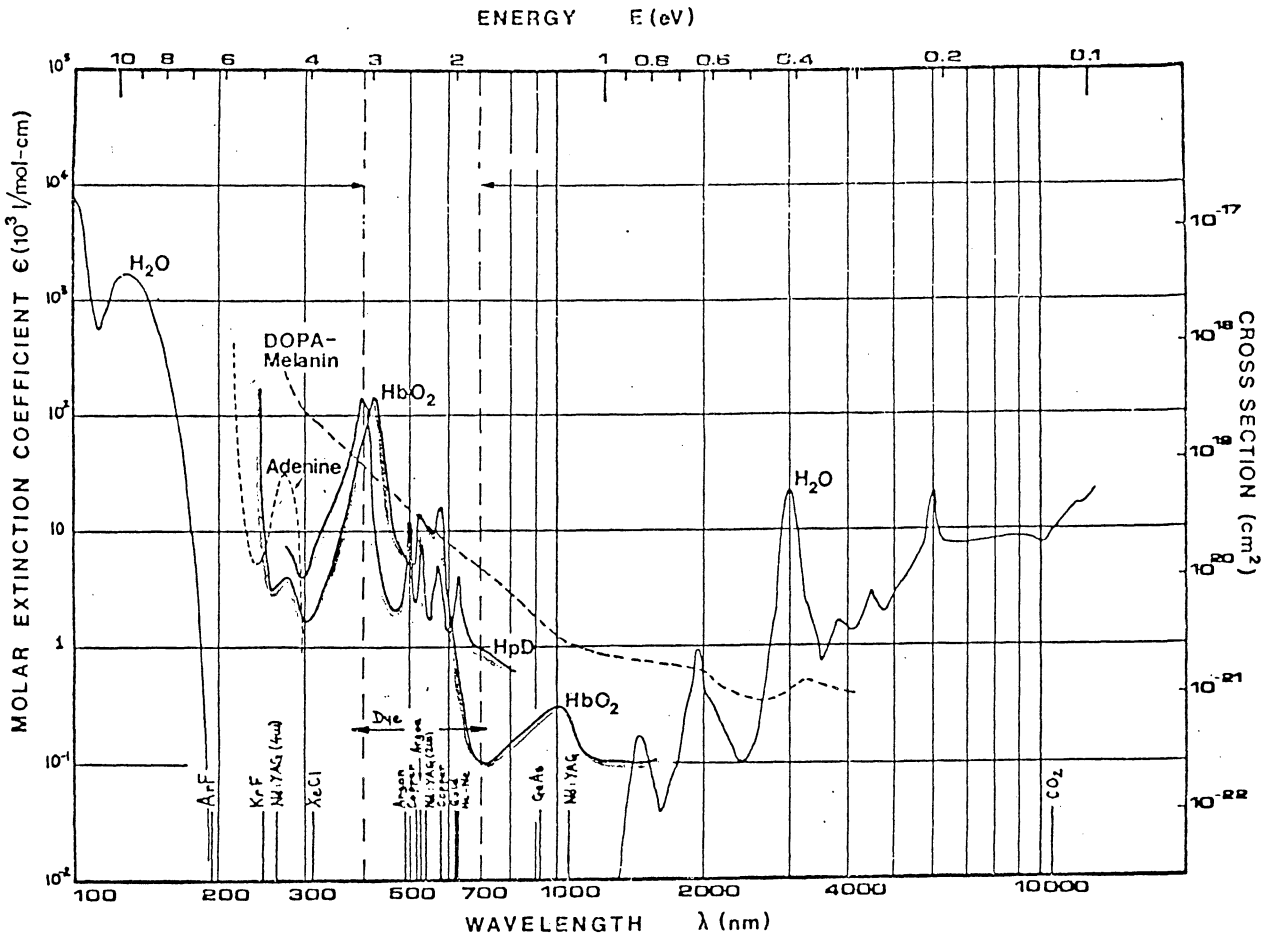



Figure 2. Molecular extinction coefficient. [3]

## 1.5 Combining treatments

By combining hyperthermia and photodynamic therapy it is possible to treat thicker tumours than with PDT alone. There may also be a synergistic effect seen in the overall kill ratio. Such effects have been observed during combined hyperthermia/ionizing radiation therapy.

When making this kind of combination it is important that both treatments work with as much efficiency as possible and without a disruptive influence on each other. For example, the water bolus used to control the surface temperature should be transparent for both laser and IR wavelengths and the termistors used to measure the temperature of the tumour should not be affected directly by the electromagnetic radiation [2].

## 2 THEORY

A one dimensional model is used as a first approximation for a theoretical deduction of the steady state temperature distribution in a tumour. This distribution can be useful when developing the regulating routine for the computer, since the temperature should be kept constant during the treatment.

The temperature change in a one dimensional model, can according to thermal conducting theory, be written as

$$\frac{1}{\chi}*\frac{\partial T}{\partial t} - \frac{\partial^2 T}{\partial x^2} = \frac{1}{\lambda}*( q_1 + \ldots + q_n )$$

where $\chi$ is the thermal diffusivity and $\lambda$ is the thermal conductivity. The q-terms are heat source or sink densities. In the proposed experiment two such densities will be present. One is the lamp which is supposed to irradiate the tumour with an intensity Io. According to diffusion theory [4] the light intensity in tissue follows Beer-Lambert's law , i.e.

$$I(x) = I_0 * e^{-\sigma x}$$

where $\sigma$ is an optical attenuation coefficient.The optical penetration depth ( OPD ) is defined as $1/\sigma$, i.e. where the light intensity has decreased by a factor of e ( OPD is depending on the wavelength of the light which means that if the heating effect of the laser in a combined hyperthermia/PDT treatment is taken into account another source term with a different OPD has to be added ). The change in intensity at a depth x is

$$\frac{\partial I}{\partial x} = -\sigma * I_0 * e^{-\sigma x}$$

This means that the tissue absorbs $\sigma*Io*e^{-\sigma x}$ which is the first source term. The second heat source ( or sink ) is the blood. If w is the blood perfusion rate (1/s), c is the specific heat of the blood ( J/kg°C ) and $\rho$ is the blood density ($kg/m^3$) then

$$q_{blood} = w*c*\rho*( T_b - T )$$

where $T_b$ is the blood temperature, can be considered the source term of the blood [1],[5]. $W*c*\rho$ is called the blood perfusion exchange factor. The complete temperature change can now be written as

$$\frac{1}{\chi}*\frac{\partial T}{\partial t} - \frac{\partial^2 T}{\partial x^2} = \frac{\sigma}{\lambda}*I(t)*e^{-\sigma x} + \frac{w*c*\rho}{\lambda}*(T_b - T) \qquad (1)$$

where I(t) is the intensity of the heat source. Because the treatment is supposed to be static (1) is solved as a steady state equation. With

$$\frac{\partial T}{\partial t} = 0 \qquad \frac{w*c*\rho}{\lambda} = \gamma$$

(1) becomes

$$\frac{\partial^2 T}{\partial x^2} = -\frac{\sigma}{\lambda} * I * e^{-\sigma x} - \gamma * (T_b - T) \qquad (2)$$

which has the limited solution

$$T(x) = T_b + A * e^{-\sqrt{\gamma} * x} - \frac{\sigma * I}{\lambda(\sigma^2 - \gamma)} * e^{-\sigma x}$$

where A is a constant. The boundary condition on the surface

$$T(0) = T_s$$

determines this constant.

$$A = T_s - T_b + \frac{\sigma * I}{\lambda(\sigma^2 - \gamma)}$$

The complete solution to (2) is

$$T(x) = T_b + \left( T_s - T_b + \frac{\sigma * I}{\lambda(\sigma^2 - \gamma)} \right) * e^{-\sqrt{\gamma} * x} - \frac{\sigma * I}{\lambda(\sigma^2 - \gamma)} * e^{-\sigma x} \qquad (3)$$

**DEV NOTE:** A program called TERMO is supplemented, which produces a graphic display of (3). To run it just load TERMO.EXE and enter *termo* on the command line. The program starts to ask for some parameters. Typical values for these are

| | |
|---|---|
| Heat intensity (I) | : 1-3 $mW/mm^2$ |
| Optical penetration depth ($1/\sigma$) | : 5-8 mm |
| Blood perfusion exchange factor ($w*c*\rho$) | : 0.007 $mW/mm^3 °C$ |
| Thermal conductivity ($\lambda$) | : 0.3-0.5 mW/mm°C |
| Blood temperature ($T_b$) | : 35-37 °C |
| Surface temperature ($T_s$) | : 20-30 °C |

The equation is drawn for 20 different surface temperatures the lowest being the one given as a parameter.
The x-axis is scaled from 0 to 12 mm and the y-axis from 30 to 52 °C with a special mark at 43 °C.
The program is documented in the program listings.
Fig 3. below shows an example of the curve.

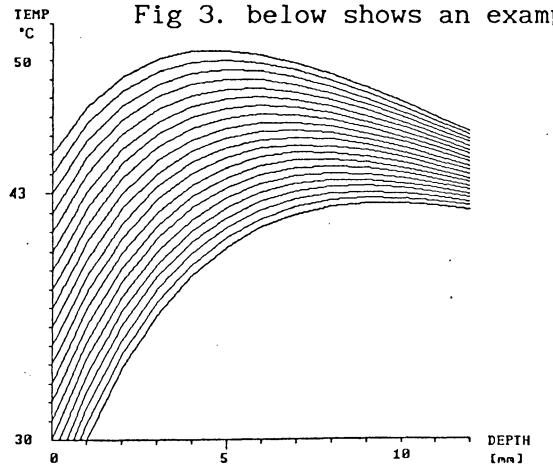
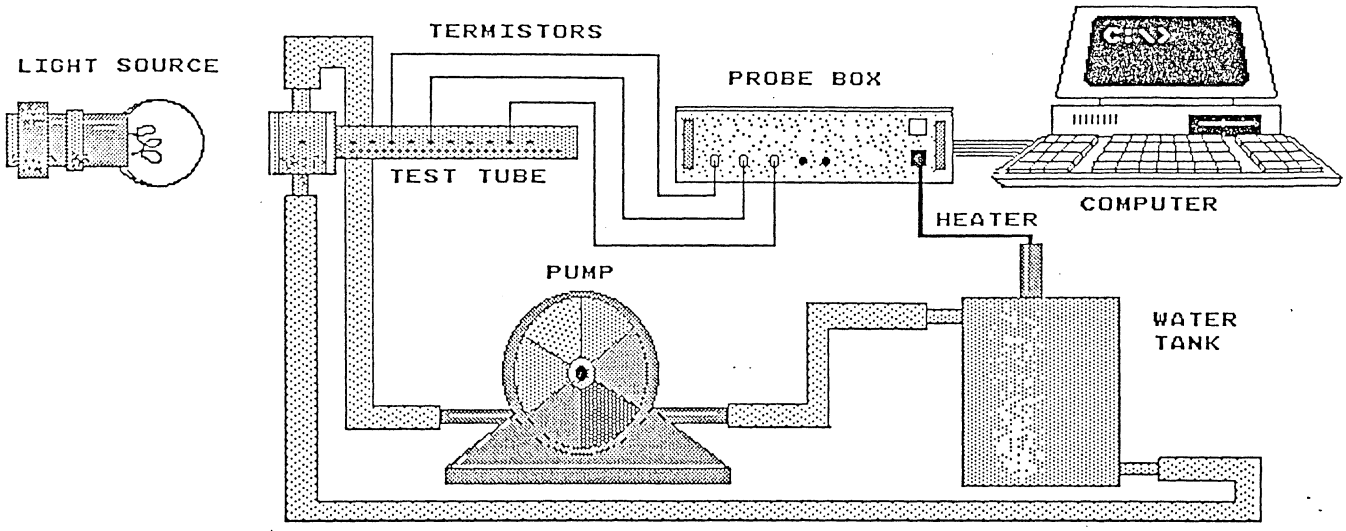



Fig 3 Sample curve from TERMO program.

# 3 EXPERIMENTAL SET UP




## 3.1 THE COOLING SYSTEM

There are two practical ways to regulate the temperature in the tumour. One is by changing the light intensity and the other by altering the surface temperature. The second alternative is the one implemented.

The surface temperature is locked by placing a water bolus on the tumour. The distilled water in the bolus is circulated by a Bi=COMET 12 V DC bi-membranes pressure pump to a container where a heater is placed. By switching the heater on the surface temperature is raised. When the heater is off the water in the system cools off to room temperature.

**DEV NOTE:** A faster cooling effect can be achieved by putting some part of the system, e.g. the container, in a cold reservoir.

Total volume of water needed in the system is 2 liters. The hoses used have inner diameters of 10 mm and 4 mm. The smaller hose is only used in connection with the test tube.

## 3.2 THE TEST TUBE

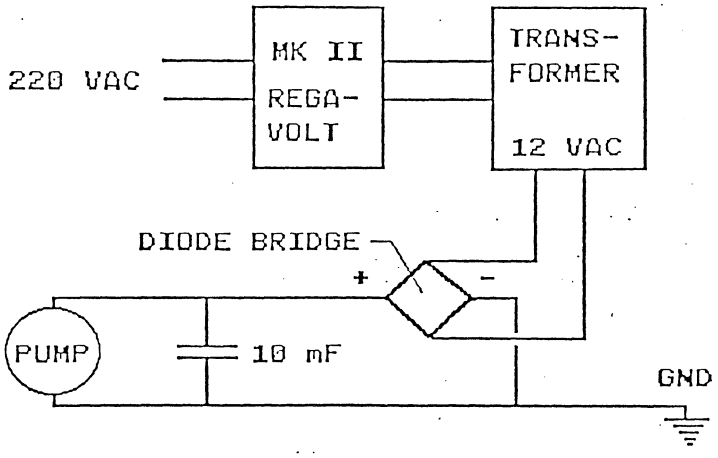
In order to test the system some sort of model for the tumour is needed. This is provided by a small metallic test tube. The small part with plastic windows and hose connections represents the water bolus. The plastic is transparent for both IR and laser (630 nm) wavelengths.

**DEV NOTE:** It may become necessary to exchange the soft plastic windows with hard ones to eliminate any focusing effects.

The tube itself can be filled with a phantom and termistors can be inserted through the holes. It may be possible to simulate blood flow by putting the tube under running water. The length of the tube is approximately 10 times the depth of a treatable tumour and the optical and thermal properties of the phantom should be chosen accordingly.

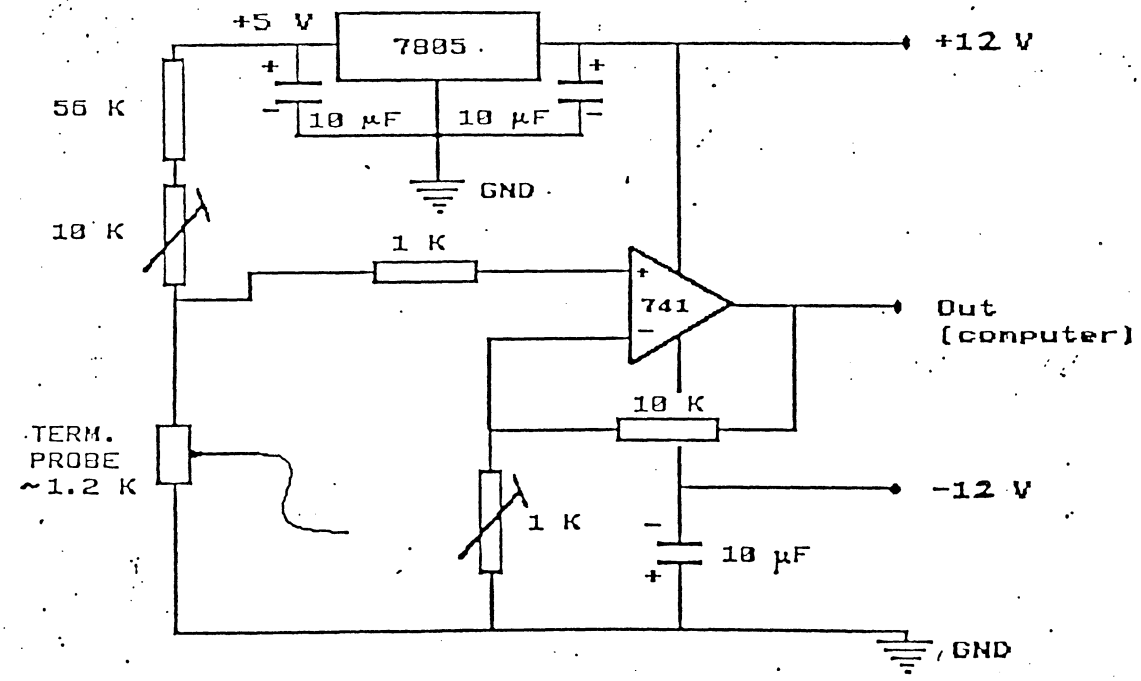
# 4 CIRCUITS

## 4.1 THE PUMP CIRCUIT




When the pump starts it requires a large current which many power supplies cannot handle. Therefore it has it's own power supply as seen above. An AC-transformer is fed by a MkII RegaVolt voltage regulator and connected to a diode bridge from which the 12 VDC required to drive the pump is taken. To maintain the voltage during switch-on a 10 mF capacitor is put parallel to the pump.

## 4.2 THE PORT AMPLIFIERS




Since the termistors basically are resistors they have to be part of a circuit in order to give information about the temperature. This information must be in a form that can be comprehended by the computer.

The computer is a Joint PC with a AD/DA-12 analog-digital/digital-analog high-performance data conversion card operating at a twelve-bit resolution. The input voltage range is 0-9 V ( adjustable ). This means that the computer is able to read a voltage between two set values and translate it to an integer between 0 and 4095 ($2^{12}$-1). This integer can then be used directly as a parameter in the programs.

The amplifier is a 741 OP with negative feedback. The 1 K variable resistor ( Rv ) determines the amplification of the voltage over the termistor ( Vterm ) as

$$V_{comp} = V_{term} * \left( 1 + \frac{10}{R_v} \right)$$

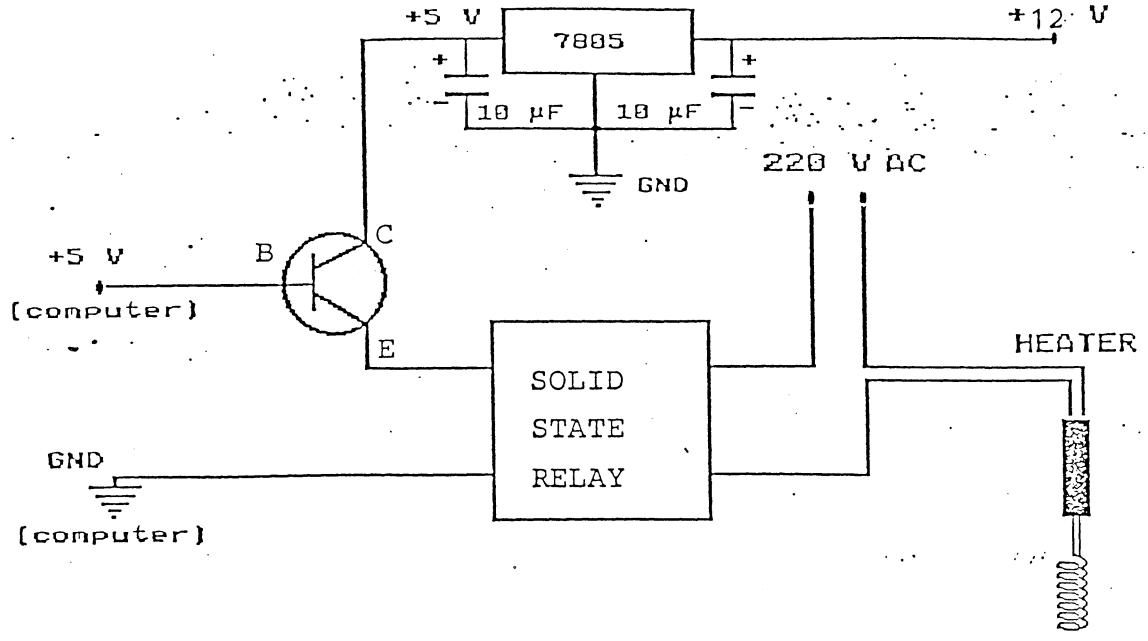
where Vcomp is the voltage read by the computer.

Two large ( 56 K and 10 K variable ) resistors are connected serially with the termistor probe to ensure that the current is approximately 100 $\mu$A. The voltage over the termistor is

$$V_{term} = \frac{5\ V}{\left( 1 + \frac{56K\Omega + R_{var}}{R_{term}} \right)}$$

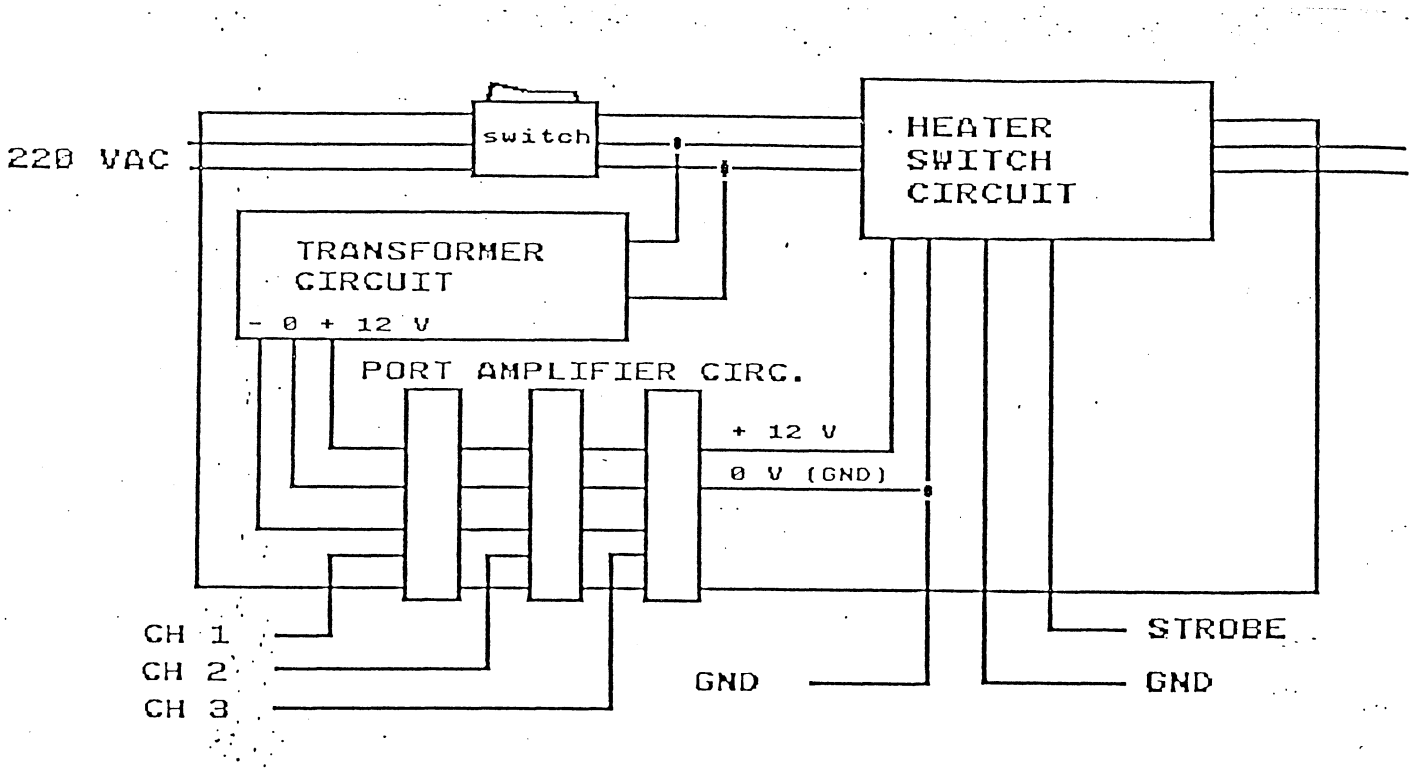
where Rvar is the resistance of the variable 10 K resistor.

## 4.3 THE HEATER SWITCH




In order to be able to switch the heater on and off from the computer a solid state relay is used. The positive input on the relay is connected to a transistor's emitter while the negative is connected directly to computer ground. The strobe output on the computers printer card is connected to the transistor's base.

4.4 THE PROBE BOX




To make hardware interfacing with the computer more simple a probe box has been constructed. The box is fed by ordinary 220 VAC. The termistors is plugged into the front of the box as is the heater. The computer cable is plugged into the rear.

Inside the box is a 220 VAC/12 VDC transformer, the heater circuit, three port amplifiers and one switch.

---

## 5 TERMISTORS & CALIBRATION

---

The termistors used are polyethylene-coated YSI 511 (Yellow Springs Inc., Model 511) with an outer diameter of 0.6 mm. They are flexible and can be inserted in the tumour through a needle. The termistors are however not unbreakable and should be handled with care. The current through a termistor should not exceed 200 µA (0.2 mA).

The calibration is based on data points stored in an array. The computer's integer representation of the voltage over the termistor at a given temperature is stored for every tenth of a deg C. When a value is read from a termistor it is rounded of to the nearest stored value and the temperature is calculated from the index in the array of that value.

The calibration is made in a water bath that cools off while being stirred.

# 6 USER'S MANUAL

## 6.1 NECESSARY FILES

These are the files that must be loaded so that the temperature control program, Hyperthermia Control, can run:

HTCTRL.EXE
HERC.BGI or EGAVGA.BGI depending on what graphic system is used.
*.CAL the files with known termistor data in them.
( this is optional since .CAL files can be created but not fetched from the program. )

## 6.2 STARTING HYPERTHERMIA CONTROL

Hyperthermia Control is run by the file HTCTRL.EXE. To start the program just enter *htctrl* on the command line.

## 6.3 THE MAIN MENU

There are only three options enabled on the main menu:

* Calibration
* Treatment
* Quit

Fig 4 below shows the structure of the menu.

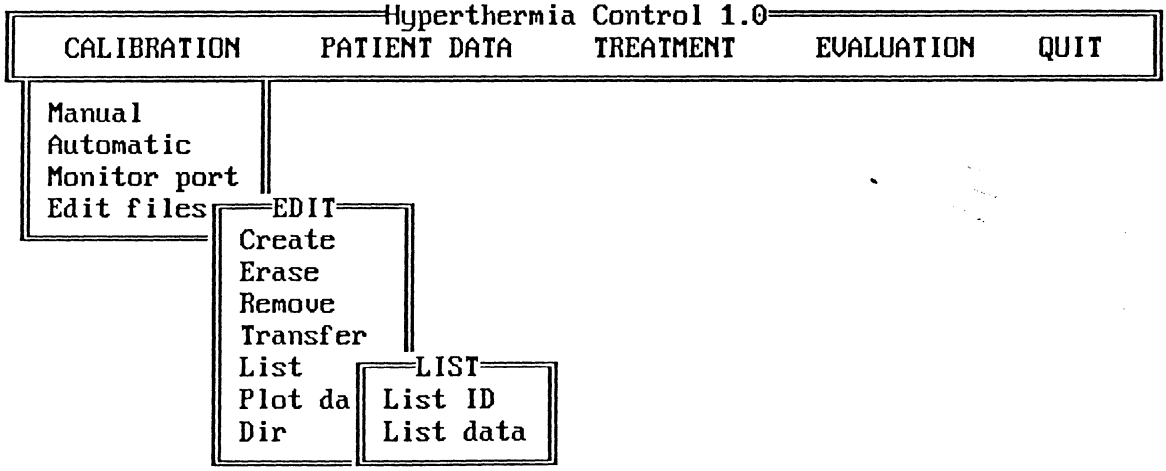



Fig 4 Example of menu structure. 'List data' under LIST chosen.

## 6.4 CALIBRATION

Before calibrating a termistor it is a good idea to prepare the file in which the calibration data are to be stored. It is not essential since a new file can be created during calibration. It is also a good idea to check the signals on the termistor ports with the 'Monitor port' option.

### 6.4.1 MANUAL CALIBRATION

When the 'Manual' option is chosen the computer asks for the number of the port on which the termistor is to calibrated. The number entered should be one of the available ports on the probe box. The computer then displays some information about the manual calibration and a menu.

The temperature of the water that the termistor is calibrated in can be raised above the upper limit of the temperature interval by using the heater. The heater is switched on by pressing [H], and [O] switches it off.

When the temperature of the water has reached the upper limit of the interval it should cool off while being stirred. If a slow thermometer (e.g an Hg-thermometer ) is used for calibration, the cooling rate should not be to high since that would probably result in a systematic error in the calibration data.

Whenever the thermometer shows the temperature displayed on the screen [Space] should be pressed. This enters the port value seen as the latest calibration point and a new temperature ( 0.1 °C lower ) is shown.

By pressing [I] the latest value is ignored and a new value can be entered by pressing [Space].

To quit calibration before finished press [Esc].
WARNING! *All data entered will be lost.*

When the last temperature is entered all the calibration data will be shown. Since the program makes sure that the calibration curve is not ambiguous ( i.e. no peaks or valleys on it ) there could be adjacent data points with the same value. These values can be separated by hand after viewing the data. When asked for it enter the data point to be changed. The old value will be shown and a new one can be entered.

After manipulation of the calibration data the program asks for the name of the file in which the termistor is to be stored. If the file entered does not exist it may be created or a new file may be specified. Then the ID of the termistor should be entered. If the ID-PS of that termistor already exists in the chosen file it may either be overwritten or a new ID could be specified.

### 6.4.2 AUTOMATIC CALIBRATION

When the 'Automatic' option is chosen the computer asks for the number of the port on which the new termistor is to calibrated. The number entered should be one of the available ports on the probe box. Next the port on which the known termistor is to be plugged in should be entered. The computer then asks for the file that contains the calibration data of the known termistor and the ID of the termistor. Then some information about the automatic calibration and a menu are displayed.

The temperature of the water that the termistor is calibrated in can be raised above the upper limit of the temperature interval by using the heater. The heater is switched on by pressing [H], and [O] switches it off.

When the temperature off the water has reached the upper limit of the interval it should cool off while being stirred. To start the calibration press [Space]. The calibration from here on is automatic.

The number in brackets is the known termistor's value for the temperature displayed, the second number is the current value read on the known termistor's port and the third number is the current value read on the unknown termistor's port.

To quit calibration before finished press [Esc].

WARNING! *All data entered will be lost.*

When the lower limit of the temperature interval is reached all the calibration data will be shown. Since the program makes sure that the calibration curve is not ambiguous ( i.e. no peaks or valleys on it ) there could be adjacent data points with the same value. These values can be separated by hand after viewing the data. When asked for it enter the data point to be changed. The old value will be shown and a new one can be entered.

After manipulation of the calibration data the program asks for the name of the file in which the termistor is to be stored. If the file entered does not exist it may be created or a new file may be specified. Then the ID of the termistor should be entered. If the ID-PS of that termistor already exists in the chosen file it may either be overwritten or a new ID could be specified.

## 6.5 MONITOR PORTS

This option allows a channel on the AD/DA card to be read. It can be used to check the signal from a termistor before calibration or when adjusting the amplification of a new port.

## 6.6 EDIT FILES

Under this heading there are several options that allow handling of the termistor files.

### Create

Creates a new termistor file. Enter the name of the file to be created. If the file already exists it can be emptied.

### Erase

Erases a termistor file. Enter the name of the file to be erased. A warning is displayed if the file is not empty.

### Remove

Removes a termistor from a file. Enter the name of the file from which the termistor is to be removed. Then enter the ID-PS number of the termistor.

Transfer

Copies a termistor from one file to another. Enter the name of the source file and the name of the destination file. Then enter the ID-PS number of the termistor.

List

The 'List' choice contains two options:

List ID

Lists the ID:s of the termistors in a file. Enter the name of the file to be listed.

List data

Lists the calibration data of a termistor. Enter the name of the file that contains the termistor. Then enter the ID-PS of the termistor. This option also allows the calibration data to be changed.

Plot data

Plots the calibration curve of a termistor. Enter the name of the file that contains the termistor. Then enter the ID-PS of the termistor. The curve that is displayed can now be altered by pressing [NumLock] and using the arrows to move the dot cursor. If [5] is pressed the dot cursors position will be the new value in that column and the cursor will move one step to the right. The new curve must be stored afterwards if the changes are to remain. The program will not allow an ambiguous curve to be stored.

Dir

Lists the names of all the *.CAL files in the current directory.

Treatment

This choice has only one option and that is

## 6.7 TEMP MEASUREMENT

Reads the temperature from three already calibrated termistors ( ID:s : 1-1, 2-2, 3-3 ). Enter the name of the file that contains the termistor data. Then enter the time interval between readings and the duration of the measurement. If the measurement is to be saved on file, enter the name of the file next ( File extension *.HTC ).

Begin measuring by pressing [Space]. The heater can be switched on and off by pressing [H] and [O] from the program. Press [Esc] to quit measurement. An example of the display is shown below [ Fig 5 ].

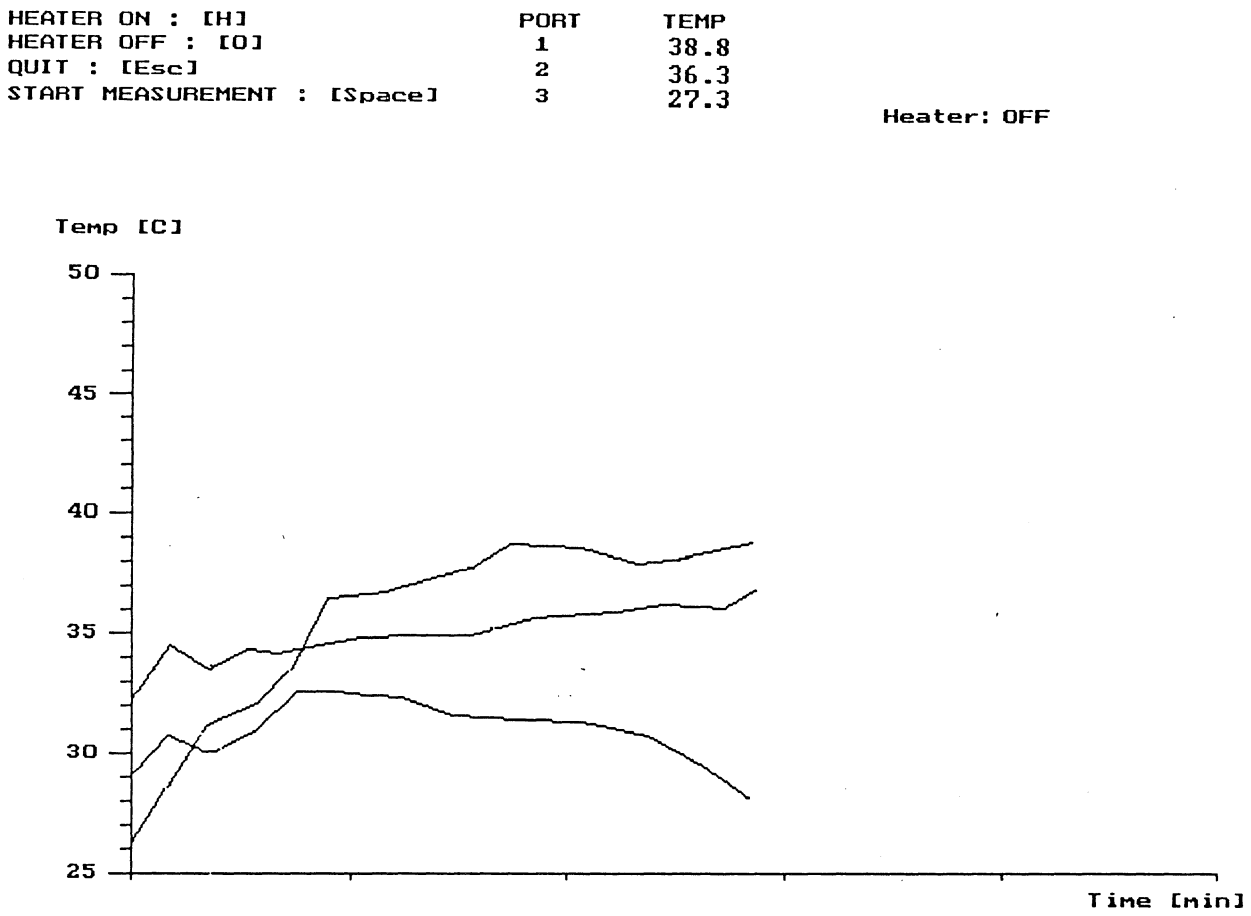



Fig 5. Temperature measurement display.

6.8 QUIT

Quits Hyperthermia Control and returns to DOS.

## 7 CONCLUSION

Using a combined treatment of hyperthermia and PDT, thicker tumours can be treated than with PDT alone. This is done by letting the PDT laser kill the superficial part of the tumour while a temperature peak of approximately 43 °C is maintained further into the tissue.

The computerized system described above is not complete, i.e. it is not fully automatic. The addition of a control routine is necessary. This routine should control the surface temperature with the heater and make adjustments based on the temperature read from termistors placed at various depths in the tumour.

It may become necessary to control the amount of irradiation also to get a stable approach to the desired temperature during the initial heating of the tumour. The hardware for such a control can be added to the probe box and run from the computer.

## 8 REFERENCES


[1] L. O. Svaasand, D. R. Doiron,T. J. Dougherty,"Temperature rise during photoradiation therapy of malignant tumors" in *Med. Phys.* 10(1) 1983 pages 11-17.

[2] P. Nilsson, "Physics and technique of microwave-induced hyperthermia in the treatment of malignant tumours", Lund 1984.

[3] S. Andersson-Engels, "Laser-induced fluorescence for medical diagnostics", Lund Reports on Atomic Physics LRAP-108 1989.

[4] L. O. Svaasand, "Optical dosimetry for direct and interstitial photoradiation therapy of malignant tumors", in *Porphyrin Localization and Treatment of Tumors, pages* 91-114, 1984 Alan R. Liss, Inc.

[5] R. Ellingsen, "Optical and thermal dosimetry during laser induced photochemical and hyperthermal treatment of cancer", University of Trondheim, Norway February 1984.

[6] K. Svanberg, "The interaction of laser light with tissue", Lund University Hospital, Lund 1989.

## Appendix A DEVELOPER'S MANUAL

The program is written in Turbo Pascal 4.0. It also uses the Turbo Pascal Professional 5.0 Toolbox for menu making. When it reads {number} it is a page reference to the Borland Turbo Pascal 4.0 manual.

In order to make the program lucid it is divided into a main program and several units. Each unit handles a certain aspect of the control system, e.g. communication with environment, calibration, graphics, menus and file editing. This makes it easy to change both conditions outside the computer and internal data representation without having to rewrite the entire program.

### 1 THE UNITS

### 1.1 The Termistor File Editing Unit ( TFEdit )

TFEdit handles the editing of the files which contain the calibration data. It is an essential unit since all termistors must be calibrated separately on each port and the idea is that the data of the termistors used in treatment is put together in one file for the program to use. This is easily accomplished by using the procedures in this unit.

#### 1.1.1 Types & Constants

A termistor is represented by an identification number and calibration data. This is done by the following type:

```
type  termtype = record
                   ID : string;
                   data : termarray;
                 end;
```

The ID is a string of unspecified length which is created during calibration only. It consists of the termistor's physical ID number, a hyphen (-) and the number of the port on which the termistor was calibrated. When the program asks about the termistor ID then the physical ID is enough but when it says ID-PS both physical ID and port specification should be entered ( with the hyphen ! ). The data are stored in an array of integer

```
type  termarray = array[0..range] of integer;
```

where range is a constant that defines the temperature interval together with the constant Maxtemp. Default values:

```
const Maxtemp = 50;    (deg C)
      range   = 250;   (1/10 deg C)
```

The lower limit of the interval becomes Maxtemp-(range/10) ( Default:25 deg C ).
WARNING! *If these constants are changed previous calibrations become useless.*

**DEV NOTE:** This can be changed with a routine that keeps old data that are in the new interval and calibrates the termistor only in the part(s) outside the old interval. Default values of the temperature interval can be initialized in the unit {64} and then changed as an option in the program.

The termistors are stored in typed files

```
type  termfile = file of termtype;
```

which lie in the directory determined by the constant

```
const  DestDir = 'C:\HTCTRL\'
```

1.1.2 Procedures

All the procedures implemented in this unit can be used by another program, i.e. they are all declared in the interface.

* ReadFile(var f:termfile;var filename:string;var found:boolean)
  reads a filename, character by character, from the keyboard ignoring everything after a dot ( including the dot ). It creates the string variable filename by adding '.cal'. Filename is then assigned to the termfile variable f. The automatic input/output error checking is turned off {530} so that when Reset tries to open the file, the boolean variable found is set to true if the operation was successful. It means that if found is true then filename is assigned to the existing open file variable f, if found is false filename is assigned to the non existing f which then have to be created.

* NoFile
  writes an error message to the screen and waits for a keyboard input. It is generally used when ReadFile has returned found as false.

* NoTerm
  writes an error message to the screen and waits for a keyboard input. It is generally used when SeekTerm has returned exist as false.

* Create(var f:termfile)
  creates and closes a file with the name previously assigned to f by ReadFile.

WARNING! *If the file already exists it is erased and replaced with a new empty file.*

* CreateCalFile
  is a complete procedure for creating new files or clearing old ones of data.

* EraseCalFile
  is a complete procedure for erasing files. It double checks if the file is not empty.

* SeekTerm(var f:termfile; termID:string; var exist:boolean;
var place:word)
seeks a termistor in f with an ID of termID. It returns exist as true if the termistor was found in the file otherwise as false. Place is the position of the termistor in the file. If exist is false then place is in end of file.

* RemoveItem(var f:termfile; place:word)
removes a termistor from the position place in the file f. It uses a file HOLD.TRF, that it creates and erases, to hold the termistors, which are not to be removed, temporarily.

* RemoveTerm
is a complete procedure for removing a termistor from a file.

* CopyTerm
is a complete procedure for copying a termistor from one file to another. It double checks against overwriting of termistors.

* ListTerm
is a complete procedure for listing the termistor ID:s of a file.

* TermDir
is a complete procedure for listing all the *.CAL files in DestDir.

* GetData(var f:termfile: place:word; var T:termarray)
copies the calibration data of the termistor in position place in the file f into the termarray variable T.

* Scroll(T:termarray)
displays the calibration data in T on the screen and allows for scrolling.

* Manipulate(var T:termarray)
allows manipulation of the calibration data in T. Should always be checked afterwards with CheckData.

* CheckData(T:termarray; var OK:boolean)
checks the calibration data in T. OK is returned as false if one or more of the calibration values are smaller than it's predecessor. In that case the calibration curve is ambiguous and should be altered with Manipulate.

* ListData
is a complete procedure for listing the calibration data of a termistor. It also allows the calibration data to be changed.

## 1.2 The Calibration Unit ( HTCal )

HTCal handles the two possible types of calibration, manual and automatic.

* AutoRead
is a complete procedure for monitoring a port on the AD/DA card. It is useful when a new port is installed or to make sure that a signal is read from a termistor before calibration.

* GetTemp(T:termarray;port:word;var temp:real;var inrange:boolean)
is a general procedure for reading the temperature on a port. T is the termarray containing the calibration data and port the number of the port to be read. Temp is the resulting temperature in deg C and inrange tells whether the port value was inside the temperature interval.

* StoreCal(T:termarray;portnr:char)
is the storing procedure in common for manual and automatic calibration. It begins by displaying the calibration data (contained in T ) on the screen ( Scroll ) and offering the possibility to change it ( Manipulate ). Then StoreCal stores the calibration data in a file asked for in the procedure. It also asks for the termistors ID and connects it with the port specification (portnr).

* ManuCal
is the complete procedure for manual calibration. It begins by asking on which port the calibration will be made.

**DEV NOTE:** If more ports are constructed their numbers must be added to the first case statement ( Line 178 ).

The calibration then proceeds as follows:
Each time [Space] is pressed the port is read and its value is stored in a termarray unless it is smaller than the last one entered. In that case the new value becomes the same as the previous. If [I] is pressed the latest value will be ignored and can be reentered. This continues until the lower limit of the temperature interval is reached. The last value cannot be ignored but it can be changed before storage by Manipulate. During the calibration the heater can be used to warm the water by pressing [H] and [O].
ManuCal ends by storing the calibration data with StoreCal.

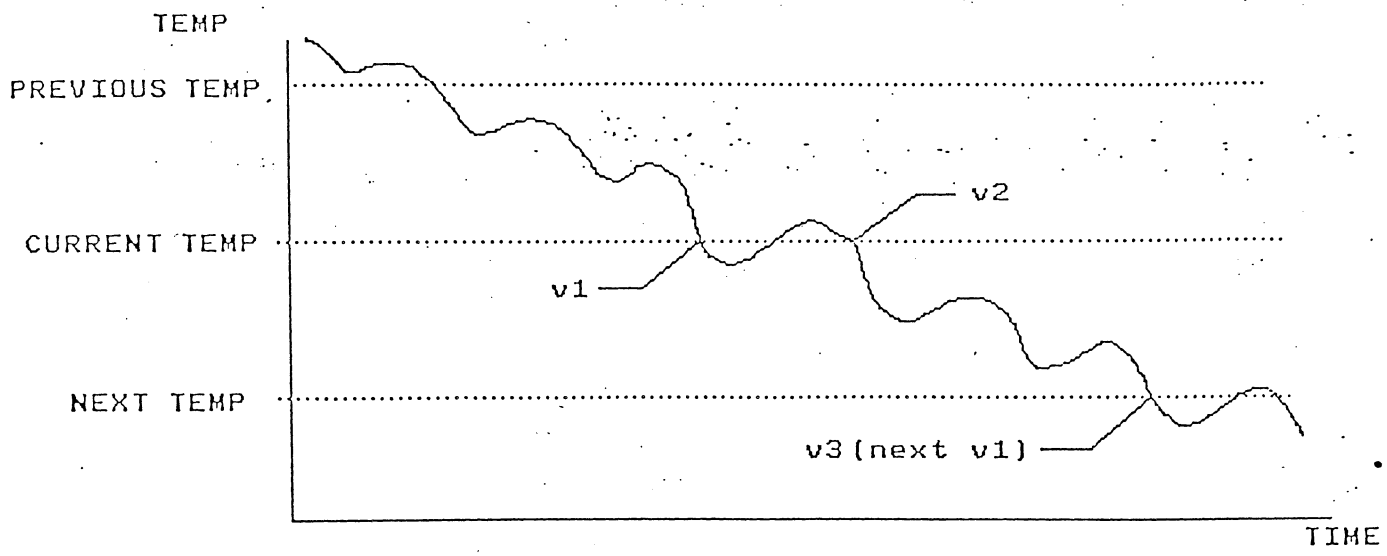
* AutoCal
is the complete procedure for automatic calibration of a termistor. It begins by asking for port numbers to be used in the calibration.

**DEV NOTE:** If more ports are constructed their numbers must be added to the first case statement ( Line 253 ).

Then it asks for the file in which the known termistor is located and its ID. The port specification does not have to be entered at this point since the program already have it. When everything is ready for calibration, i.e. the ports are defined and the known termistor data have been found, AutoCal waits for [Space] to be pressed. The value read from the known termistor is checked to see if the upper limit of the temperature interval has been reached. If so then both variables v1 and v2 are assigned the value read from the unknown termistor. AutoCal then enters a loop which continues until the lower limit of the temperature interval is reached or [Esc] is pressed.

The principle of the calibration is the following: When a new temperature ( 0.1 °C lower ) is reached, i.e. the known termistor reaches a new calibration point, the mean value of the two variables v1 and v2 is taken and stored as a calibration value for the unknown termistor's previous temperature. Then the value read from the unknown termistor when the new temperature was reached is assigned to v1 and v2 ( the variable v3 serves as a temporary memory so that both termistors are read as simultaneously as possible ). If the temperature read by the known termistor should rise above the current temperature again the variable *above* is set to TRUE. This means that when the lower temperature is reached again, the value

read from the unknown termistor is assigned to v2. This is why it is the mean value of v1 and v2 that is stored. The principle is shown in a diagram below.




## 1.3 The Communication Unit ( Commo )

Commo handles the communication with the environment, e.g. the keyboard and the probe box.

Three constants are defined in this unit:

```
const  adport = $270;
       prctrlport = $3BE;
       readprec = 100;
```

The first constant, adport, is the address of the computers I/O memory area where the AD/DA card is located. The second, prctrlport, is the address in the I/O memory area for the strobe pin on the computers printer card. The third, readprec, is the number of conversions to be done by the AD/DA card.

* WaitKey
  is general procedure that halts program execution until a key is pressed on the keyboard.

* ReadPortNr(var port:word)
  is a general procedure that asks for a port number between 0 and 15.

* ReadPort(portnr:integer):integer
  is a general function for reading a channel on the AD/DA card. ReadPort takes a mean value over *readprec* readings. A reading begin by selecting the channel and clearing the register ( offset 3 ). Then offsets 4 and 5 are looped back seven times to start the high and low 6-bit conversions. After that the results of the conversions are read from offsets 1 and 2 ( low and high word ) and added.

* HighOut
  is a general procedure for putting the strobe pin on the printer card high ( by assigning it zero ; inverse logic on the strobe ).

* LowOut
  is a general procedure for putting the strobe pin on the printer card high ( by assigning it one ; inverse logic on the strobe ).

**DEV NOTE :** It may be possible to use the DA output on the AD/DA card as a source for switching the heater on/off instead of the strobe on the printer card.

1.4 The Menu Unit ( HTMenu )

HTMenu contains the source code for the main menu and its submenues.

* InitMenu(var M:Menu)
  initializes the menu identified by M.

* ShowMenu(var M:Menu)
  is a redefinition of ReDrawMenu so that the main program does not have to use TPMenu. ShowMenu redraws the menu M as it was last displayed.

* Choice:MenuKey
  is a redefinition of MenuChoice so that the main program does not have to use TPMenu. Choice returns the menu choice as an integer value. That value is the fourth parameter in a MenuItem statement.

There are two ways to change the menu; by changing the source code in HTMENU.PAS or by using the MAKEMENU program.
For small changes, like a new help line for an existing item, it's easiest to alter the source code directly in the HTMENU.PAS file and recompile.
If the menu is to be changed a lot do the following:

* Run MAKEMENU.EXE.
* Choose 'File' on the main menu.
* Choose the 'Read lib' option on the File sub menu.
* Enter *latmenu* as the menu library.
* Enter *latmenu* as the menu library ID.
* Make the changes of the menu by using the various options in MAKEMENU.
* Choose the 'Write lib' option on the File sub menu.
* Choose the 'Generate source' option on the File sub menu.
* Quit MAKEMENU.
* Use the Turbo Pascal Editor to replace the InitMenu procedure in HTMENU.PAS with the InitMenu procedure in LATMENU.PAS.
* Recompile the main program with MAKE.

1.4 The Graphics Unit ( HTGraph )

This unit contains procedures for graphic representation of calibration curves. Graphic procedures for treatment and experiments can also be placed here. A makeshift routine for temperature measurements is also placed here.

* InitGraphics
  is a general procedure for initializing the graphics on the current drive. It should only be called once by the main program preferably in the beginning.

* TermPlot
  is a complete procedure for plotting the calibration curve of a termistor. It also allows a rough manipulation of the curve.

**DEV NOTE** : A modified TermPlot could serve as a procedure to pick out good points for an equation-based calibration.

* HeatCtrl
  is a makeshift routine that uses the three termistors 1-1, 2-2 and 3-3 specifically in order to measure the temperature. It is not recommended as a basis for further development due to its lack of structure.

## 2 THE MAIN PROGRAM (HTCtrl)

The main program consists of a simple loop which polls the menu for a choice. Depending on the choice made, a complete procedure in one of the units is called. After each choice and when the loop is exited the heater is turned off to be on the safe side.

```
 1: ä ************************************************** å
 2: ä *                                                * å
 3: ä *            Hyperthermia Control                * å
 4: ä *                Main Program                    * å
 5: ä *                                                * å
 6: ä ************************************************** å
 7:
 8: program HTControl;
 9: ägn+å
10:
11: uses TPCrt,
12:      HTMenu,
13:      Commo,
14:      TFEdit,
15:      HTCal,
16:      HTGraph;
17:
18: ä* MAIN PROGRAM *å
19:
20: begin
21:  InitMenu(M);                                              ä HTMenu å
22:  repeat
23:   LowOut;          ä Make sure that heater is off å        ä Commo å
24:   ClrScr;
25:   ShowMenu(M);                                             ä HTMenu å
26:   key:=Choice;                                             ä HTMenu å
27:   ClrScr;
28:   case key of                                 ä MENU ITEM å
29:      4: ;                                       ä QUIT å
30:      5: ManuCal;                               ä Manually å    ä HTCal å
31:      6: AutoCal;                              ä Automatic å    ä HTCal å
32:     11: HeatCtrl;                          ä Temp measurement å ä HTGraph å
33:     14: CreateCalFile;                         ä Create å      ä TFEdit å
34:     15: EraseCalFile;                          ä Erase å       ä TFEdit å
35:     16: RemoveTerm;                            ä Remove å      ä TFEdit å
36:     17: CopyTerm;                               ä Copy å       ä TFEdit å
37:     24: ListTerm;                              ä List ID å     ä TFEdit å
38:     23: TermDir;                                ä Dir å        ä TFEdit å
39:     26: TermPlot;                             ä Plot data å    ä HTGraph å
40:     25: ListData;                              ä List data å   ä TFEdit å
41:     27: AutoRead;                            ä Monitor port å   ä HTCal å
42:   end;
43:  until key=4;
44:  LowOut;           ä Make sure that heater is off å          ä Commo å
45: end.
```

```
ä ************************************************
  *                                              *
  *                 UNIT HTMenu                  *
  *                                              *
  *    This unit contains procedures and functions   *
  *         for displaying the main menu.        *
  *                                              *
  ************************************************ å

unit HTMenu;

interface

uses
  TPString,
  TPCrt,
  TPCmd,
  TPWindow,
  TPMenu;

var
  M : Menu;
  Ch : Char;
  Key : MenuKey;

procedure InitMenu(var M : Menu);
ä Initiates the main menu å

procedure ShowMenu(var M : Menu);
ä Displays the main menu as it was last seen å

function Choice:MenuKey;
ä Returns the the menu choice as an integer value å

ä-------------------------------------------------------------------------å

implementation

procedure InitMenu(var M : Menu);
const
  Color1 : MenuColorArray = (¤0E, ¤2E, ¤03, ¤78, ¤09, ¤0E, ¤08, ¤78);
  Frame1 : FrameArray = '      ';
  Frame2 : FrameArray = '      ';

begin
  äCustomize this call for special exit characters and custom item displayså
  M := NewMenu(ÄÅ, nil);

  SubMenu(3,7,24,Horizontal,Frame1,Color1,'Hyperthermia Control 1.0');
    MenuMode(False, True, False);
    MenuWidth(73);
    MenuItem('CALIBRATION',4,1,1,'Calibrate new termistor');
    SubMenu(4,9,24,Vertical,Frame2,Color1,'');
      MenuMode(False, True, False);
      MenuItem('Manual',1,1,5,'Manual input using Hg-thermometer');
      MenuItem('Automatic',2,1,6,'Automatic input using known termistor');
      MenuItem('Monitor port',3,9,27,'Monitor a port on the ADDA card');
      MenuItem('Edit files',4,1,13,'Edit termistor files,create new ones etc.');
      SubMenu(16,13,24,Vertical,Frame1,Color1,'EDIT');
        MenuMode(False, True, False);
        MenuItem('Create',1,1,14,'Create new termistor file');
        MenuItem('Erase ',2,1,15,'Erase a termistor file from directory');
        MenuItem('Remove ',3,1,16,'Remove a termistor from a termistor file');
        MenuItem('Transfer',4,1,17,'Transfer a termistor from one file to anothe
        MenuItem('List',5,1,22,'List all the termistor IDs in a file');
        SubMenu(25,18,24,Vertical,Frame1,Color1,'LIST');
          MenuMode(False, True, False);
          MenuItem('List ID',1,6,24,'List the termistor ID:s in a file');
          MenuItem('List data',2,1,25,'List the calibration data of a termistor'
          PopSublevel;
          MenuItem('Plot data',6,1,26,'Plot calibration data of a termistor');
          MenuItem('Dir',7,1,23,'List all *.CAL files in directory');
          PopSublevel;
        PopSublevel;
    MenuItem('PATIENT DATA',20,1,2,'Enter ID and notes about the patient');
    SubMenu(25,9,24,Vertical,Frame2,Color1,'');
      MenuMode(False, True, False);
      MenuItem('ID',1,1,18,'Enter an identification number for the patient to be
      MenuItem('Notes',2,1,19,'Enter optional information about the patient');
      PopSublevel;
    MenuItem('TREATMENT',37,1,3,'Set up and begin treatment');
    SubMenu(35,9,24,Vertical,Frame2,Color1,'');
      MenuMode(False, True, False);
      MenuHeight(5);
      MenuItem('Temp measurement',2,1,11,'Start temperature measurement');
      PopSublevel;
    MenuItem('EVALUATION',51,1,12,'Evaluation of treatment');
    SubMenu(54,9,24,Vertical,Frame2,Color1,'');
      MenuMode(False, True, False);
      MenuItem('List',1,1,20,'List temperatures for all termistors during treatm
      MenuItem('Graph',2,1,21,'Draw graph showing temperatures for all termistor
      PopSublevel;
    MenuItem('QUIT',65,1,4,'Quit Hyperthermia Control');
    PopSublevel;

  ResetMenu(M);
end;
ä End InitMenu å

procedure ShowMenu(var M : Menu);
begin
  ReDrawMenu(M);
end;
ä End ShowMenu å

function Choice:MenuKey;
begin
  Choice:=MenuChoice(M,Ch);
end;
ä End Choice å

end.
ä End HTMenu å
```

```
ä ******************************************************
  *                                                    *
  *                     UNIT Commo                     *
  *                                                    *
  *      This unit contains procedures and functions   *
  *    for computer communication with the environment.*
  *                                                    *
  ****************************************************** å

unit commo;

interface

   uses TPCrt;

   const  adport=¤270;
          prctrlport=¤3BE;
          readprec=100;

   procedure WaitKey;
   ä Halts program execution until a key on the keyboard is pressed å

   procedure ReadPortNr(var port:word);
   ä Reads and returns a port number from the keyboard å

   function ReadPort(portnr:integer):integer;
   ä Reads a channel (portnr) on the ADDA card
     and returns it as an integer mean value å

   procedure HighOut;
   ä Puts strobe pin on printer output high ( inverse logic ) å

   procedure LowOut;
   ä Puts strobe pin on printer output low ( inverse logic ) å

ä-------------------------------------------------------------------------å

implementation


procedure WaitKey;
var dumpchar : char;

begin
  repeat ;
  until KeyPressed;
  dumpchar:=ReadKey;
end;
ä End WaitKey å

procedure ReadPortNr(var port:word);
begin
  repeat
   writeln('Portnr must be an integer between 0 and 15.');
   write('Portnr: ');
  ä¤I-å              ä Turn off automatic input error checking å
   readln(port);
  ä¤I+å              ä Turn on automatic input error checking å
  until (IOResult=0) and (port<=15) and (port>=0);
end;
ä End ReadPortNr å

function ReadPort(portnr:integer):integer;
  var  n,k,adhigh,adlow,dummy : integer;
       sum : longint;

 begin
  sum:=0;
  for k:=1 to readprec do    ä Mean value over readprec readings å
   begin
    portÄadportÅ:=portnr;
    dummy:=portÄadport+3Å;
    n:=7;
    repeat
      dummy:=portÄadport+4Å;
      n:=pred(n);
    until n=0;
    n:=7;
    repeat
      dummy:=portÄadport+5Å;
      n:=pred(n);
    until n=0;
    adhigh:=portÄadport+2Å;
    adhigh:=adhigh and ¤F;
    adhigh:=adhigh shl 8;
    adlow:=portÄadport+1Å;
    sum:=sum+(adhigh or adlow);
   end;
  readport:=round(sum/readprec);
 end;
ä End ReadPort å

procedure HighOut;
  begin
   portÄprctrlportÅ:=0;    ä Inverse logic å
  end;
ä End HighOut å

procedure LowOut;
  begin
   portÄprctrlportÅ:=1;    ä Inverse logic å
  end;
ä End LowOut å

end.
ä End Commo å
```

```
 1: ä ************************************************
 2:   *                                              *
 3:   *                 UNIT TFEdit                  *
 4:   *                                              *
 5:   *    This unit contains procedures for editing  *
 6:   *          of the termistor files.             *
 7:   *                                              *
 8:   ************************************************ å
 9:
10: unit TFEdit;
11:
12: interface
13:
14: uses TPCrt,dos,commo;
15:
16: const MaxTemp=50;   ä Maxtemp defines the upper limit of the termistor interval
17:       range=250;    ä Range must be 10*(temperature interval in deg C) å
18:       DestDir='C:ÖHTCTRLÖ';  ä Directory to store termistor files in å
19:
20: type  termarray=arrayÄ0..rangeÅ of integer;
21:       termtype=record
22:                  ID : string;
23:                  data : termarray;
24:                end;
25:       termfile=file of termtype;
26:
27: procedure ReadFile(var f:termfile;var filename:string; var found:boolean);
28: ä Assigns and open file filename if it exists å
29:
30: procedure NoFile;
31: ä Error message for use when file not found å
32:
33: procedure NoTerm;
34: ä Error message for use when termistor not found å
35:
36: procedure Create(var f:termfile);
37: ä Creates and closes an file å
38:
39: procedure CreateCalFile;
40: ä Complete procedure for creating a termistor file å
41:
42: procedure EraseCalFile;
43: ä Complete procedure for erasing a termistor file å
44:
45: procedure SeekTerm(var f:termfile;termID:string;var exist:boolean;
46:                    var place:word);
47: ä Searches for termID in file f.Returns place in file if it exists å
48:
49: procedure RemoveItem(var f:termfile;place:word);
50: ä Removes termistor in place from file f å
51:
52: procedure RemoveTerm;
53: ä Complete procedure for removing a termistor from a file å
54:
55: procedure CopyTerm;
56: ä Complete procedure for copying a termistor from one file to another å
57:
58: procedure ListTerm;
59: ä Complete procedure for listing the termistor ID:s in a file å
60:
61: procedure TermDir;
62: ä Complete procedure for listing the .CAL files in DestDir å
63:
64: procedure GetData(var f:termfile;place:word;var T:termarray);
65: ä Returns the termarray T from place in file f å
66:
67: procedure Scroll(T:termarray);
68: ä Displays and scrolls the termarray T å
69:
70: procedure Manipulate(var T:termarray);
71: ä Enables manipulation of termistor data in T å
72:
73: procedure CheckData(T:termarray;var OK:boolean);
74: ä Checks termistor data in T against ambiguity å
75:
76: procedure ListData;
77: ä Complete procedure for listing a termistor å
78:
79: ä-------------------------------------------------------------------------å
80: implementation
81:
82: procedure ReadFile(var f:termfile;var filename:string;var found:boolean);
83: var ignore : boolean;
84:     chr : char;
85: begin
86:   filename:='';
87:   repeat
88:     ignore:=false;
89:     found:=false;
90:     while not eoln do
91:      begin
92:       read(chr);
93:       if chr='.' then
94:         ignore:=true;
95:       if not ignore then
96:         filename:=ConCat(filename,chr);
97:      end;
98:     readln;
99:   until filename<>'';                ä Must have a filename å
100:  writeln;
101:  filename:=ConCat(filename,'.cal');
102:  Assign(f,Concat(DestDir,filename));
103:  ä¤I-å      ä Turn off automatic input/output error checking å
104:  Reset(f);
105:  ä¤I+å      ä Turn on automatic input/output error checking å
106:  if IOResult=0 then
107:    found:=true;
108: end;
109: ä End ReadFile å
110:
111: procedure NoFile;
112: begin
113:    writeln('File did not exist.Press any key to return to menu.');
114:    WaitKey;                                          ä Commo å
115: end;
116: ä End NoFile å
117:
118: procedure NoTerm;
119: begin
120:   writeln('Termistor did not exist in that file.');
121:   writeln('Press any key to return to menu.');
122:   WaitKey;                                           ä Commo å
123: end;
124: ä End NoTerm å
125:
126: procedure Create(var f:termfile);
127: begin
128:  Rewrite(f);
129:  Close(f);
130: end;
131: ä End Create å
132:
133: procedure CreateCalFile;
134: var   created,found : boolean;
135:       filename : string;
136:       f : termfile;
137: begin
138:  repeat
139:   created:=true;
140:   writeln('Enter the name of the file.( *.CAL )');
```

```
141:   ReadFile(f,filename,found);
142:   if found then
143:     begin
144:      writeln('File already exists.Do you want to clear it of data ? ÄY/*Å');
145:      case ReadKey of
146:        'Y','y' : Create(f);
147:      else
148:        created:=false;
149:      end;
150:     end
151:   else
152:     Create(f);
153:  until created;
154:  writeln('File ',filename,' created. Press any key to return to menu.');
155:  WaitKey;                                          ä Commo å
156: end;
157: ä End CreateCalFile å
158:
159: procedure EraseCalFile;
160: var f : termfile;
161:     filename : string;
162:     OK,found : boolean;
163:
164: begin
165:   writeln('Enter the name of the file to be erased.');
166:   ReadFile(f,filename,found);
167:   if not found then
168:     NoFile
169:   else
170:     begin
171:       OK:=true;
172:       if FileSize(f)<>0 then
173:         begin
174:           writeln('File is not empty. Are you sure you want to erase it ? ÄY/*Å'
175:           case ReadKey of
176:             'Y','y' : ;
177:           else
178:             OK:=false;
179:           end;
180:         end;
181:       if OK then
182:         begin
183:           Close(f);
184:           Erase(f);
185:           writeln('File erased.Press any key to return to menu.');
186:           WaitKey;                                    ä Commo å
187:         end;
188:     end;
189: end;
190: ä End EraseCalFile å
191:
192:
193: procedure SeekTerm(var f:termfile; termID:string; var exist:boolean;
194:                    var place:word);
195: var  k : integer;
196:      term : termtype;
197:
198: begin
199:   Reset(f);
200:   exist:=false;
201:   place:=FileSize(f);  ä If element does not exist place is in end of file å
202:   k:=1;
203:   while (k<=FileSize(f)) and (not exist) do
204:     begin
205:       k:=succ(k);
206:       read(f,term);
207:       if term.ID=termID then
208:         begin
209:           exist:=true;
210:           place:=pred(FilePos(f));  ä File pointer is behind the read element å
211:         end;
212:     end;
213: end;
214: ä End SeekTerm å
215:
216:
217: procedure RemoveItem(var f:termfile;place:word);
218: var k,MaxPlace : word;
219:     term : termtype;
220:     holdf : termfile;
221:
222: begin
223:   Reset(f);
224:   Assign(holdf,ConCat(DestDir,'hold.trf'));
225:   Rewrite(holdf);
226:   MaxPlace:=pred(FileSize(f));   ä Last element before end of file å
227:   for k:=0 to MaxPlace do
228:     begin
229:       read(f,term);
230:       if k<>place then       ä Do not copy term on place, it is to be removed å
231:         write(holdf,term);
232:     end;
233:   Rewrite(f);
234:   Reset(holdf);
235:   while not EOF(holdf) do
236:    begin
237:     read(holdf,term);
238:     write(f,term);
239:    end;
240:   Close(holdf);
241:   Erase(holdf);
242: end;
243: ä End RemoveItem å
244:
245:
246:
247: procedure RemoveTerm;
248: var  f:termfile;
249:      filename,termID:string;
250:      exist,found:boolean;
251:      place:word;
252:
253: begin
254:   writeln('Enter name of the file from which you want to remove termistor');
255:   ReadFile(f,filename,found);
256:   if not found then
257:     NoFile
258:   else
259:     begin
260:       writeln('Enter the ID-PS number of the termistor to be removed');
261:       readln(termID);
262:       SeekTerm(f,termID,exist,place);
263:       if not exist then
264:         NoTerm
265:       else
266:         begin
267:           RemoveItem(f,place);
268:           writeln('Termistor removed. Press any key to return to menu.');
269:           WaitKey;
270:         end;
271:       Close(f);
272:     end;
273: end;
274: ä End RemoveTerm å
275:
276:
277: procedure CopyTerm;
278: var f1,f2 : termfile;
279:     filename1,filename2,termID : string;
280:     place1,place2 : word;
```

```
    exist1,exist2,OK,found : boolean;
    hold : termtype;

begin
  write('From file : ');
  ReadFile(f1,filename1,found);
  if not found then
    NoFile
  else
    begin
      write('To file : ');
      ReadFile(f2,filename2,found);
      OK:=true;
      if not found then
        begin
          writeln('File did not exist. Do you want to create it ? ÄY/*Å');
          case ReadKey of
            'Y','y' : Create(f2);
          else
            OK:=false;
          end;
        end;
      if OK then        ä File to write to exists å
        begin
          write('Termistor ID-PS :');
          readln(termID);
          SeekTerm(f1,termID,exist1,place1);
          if not exist1 then
            begin
              OK:=false;
              Close(f2);
              NoTerm;
            end
          else
            begin
              SeekTerm(f2,termID,exist2,place2);
              if exist2 then
                begin
                  writeln('Termistor already exists in ',filename2);
                  writeln('Do you want to write over ? ÄY/*Å');
                  case ReadKey of
                    'Y','y' :  RemoveItem(f2,place2);
                  else
                    begin
                     OK:=false;
                     Close(f2);
                    end;
                  end;
                end;
            end;
        end;
      if OK then
        begin
          Reset(f1);
          Seek(f1,place1);
          read(f1,hold);
          Reset(f2);
          Seek(f2,FileSize(f2));
          write(f2,hold);
          Close(f2);
          writeln('Termistor ',termID,' copied from ',filename1,' to ',filename2
          writeln('Press any key to return to menu');
          WaitKey;
        end;
      Close(f1);
    end;
end;
ä End CopyTerm å


procedure ListTerm;
var  f : termfile;
     filename : string;
     term : termtype;
     found : boolean;

begin
  writeln('Enter the name of the file that you want listed.');
  ReadFile(f,filename,found);
  if not found then
    NoFile
  else
    begin
      writeln('Termistors in ',filename,':');writeln;
      while not Eof(f) do
        begin
          read(f,term);
          writeln('      ',term.ID);
        end;
      writeln;
      writeln('Press any key to return to menu.');
      WaitKey;
    end;
end;
ä End ListTerm å


procedure TermDir;
var frec : SearchRec;

begin
  ClrScr;
  writeln('These are the .CAL files in ',DestDir);writeln;
  FindFirst(ConCat(DestDir,'*.cal'),anyfile,frec);
  while ( DosError=0 ) do
   begin
    writeln(frec.name);
    FindNext(frec);
   end;
  writeln;
  writeln('Press any key to return to menu.');
  WaitKey;
end;
ä End TermDir å


procedure GetData(var f:termfile;place:word;var T:termarray);
var  term : termtype;

begin
 Seek(f,place);
 read(f,term);
 T:=term.data;
end;
ä End GetData å


procedure Scroll(T:termarray);
var k,j : integer;

begin
 ClrScr;
 writeln('This is the calibration data.');
 writeln('Press ÄFÅ to scroll forward, ÄBÅ to scroll backward and');
 writeln('ÄEscÅ to quit scrolling.');
 Window(1,4,80,25);
 for k:=0 to (range-20) do
    begin
      for j:=k to (k+20) do
        writeln('Datapoint ',j,': ',TÄjÅ);
```

```
421:       case ReadKey of
422:         #¤1B : k:=range-20;
423:         'F','f' :  if k=range-20 then k:=-1;
424:         'B','b' :  if k>0 then k:=k-2 else k:=range-21;
425:       else
426:         k:=k-1;
427:       end;
428:       ClrScr;
429:     end;
430:  Window(1,1,80,25);
431:  ClrScr;
432: end;
433: ä End Scroll å
434:
435:
436: procedure Manipulate(var T:termarray);
437: var OK : boolean;
438:     k : integer;
439: begin
440:  repeat
441:    OK:=true;
442:    writeln('Enter the number of the data point you want to change.');
443:   ä¤I-å  ä Turn off automatic input error checking å
444:    repeat
445:      writeln('Datapoint must be an integer between 0 and ',range);
446:      write('Datapoint: ');
447:      readln(k);
448:    until (IOResult=0) and (k<=range) and (k>=0);
449:    writeln('Old value: ',TÄkÅ);
450:    writeln('Enter the new value.');
451:    repeat
452:      writeln('Value must be an integer between 0 and 4095.');
453:      write('New value: ');
454:      readln(TÄkÅ);
455:    until (IOResult=0) and (TÄkÅ<=4095) and (TÄkÅ>=0);
456:   ä¤I+å  ä Turn on automatic input error checking å
457:     writeln('Do you want to change more ? ÄY/*Å');
458:     case ReadKey of
459:       'Y','y' : OK:=false;
460:     end;
461:   until OK;
462: end;
463: ä End Manipulate å
464:
465: procedure CheckData(T:termarray;var OK:boolean);
466: var  k : integer;
467:
468: begin
469:  OK:=true;
470:  for k:=1 to range do
471:   if  TÄkÅ < TÄk-1Å then
472:     begin
473:       writeln('Data point ',k-1,' or ',k,' wrong.');
474:       OK:=false;
475:     end;
476:  if not OK then
477:    writeln('Ambigious data. Change the above data points.');
478:  writeln('Press any key to continue.');
479:  WaitKey;
480: end;
481: ä End CheckData å
482:
483: procedure ListData;
484: var f : termfile;
485:     term : termtype;
486:     T : termarray;
487:     filename,termID : string;
488:     exist,OK,found : boolean;
489:     place : word;
490:
491: begin
492:   writeln('Enter name of the file in which the termistor exists.');
493:   ReadFile(f,filename,found);
494:   if not found then
495:     NoFile
496:   else
497:     begin
498:       writeln('Enter the ID-PS number of the termistor to be listed');
499:       readln(termID);
500:       SeekTerm(f,termID,exist,place);
501:       if not exist then
502:         NoTerm
503:       else       ä Both File And Termistor Exist å
504:         begin
505:          GetData(f,place,T);           ä Get Termistor From File å
506:          RemoveItem(f,place);          ä Remove Termistor. No Doubles å
507:          repeat
508:           Scroll(T);                   ä Shows Calibration Data å
509:           writeln('Do you want to manipulate the data? ÄY/*Å');
510:           case ReadKey of
511:             'Y','y' : Manipulate(T);   ä Allows Change Of Data å
512:           end;
513:           CheckData(T,OK);             ä Checks If Data Are Correct å
514:          until OK;                     ä Data Correct å
515:          term.ID:=termID;
516:          term.data:=T;
517:          Seek(f,FileSize(f));
518:          write(f,term);                ä Write Termistor To File å
519:         end;
520:       Close(f);
521:     end;
522: end;
523: ä End ListData å
524:
525: end.
526: ä End TFEdit å
```

```
ä ************************************************
  *                                              *
  *                UNIT HTGraph                  *
  *                                              *
  *   This unit contains procedures for graphic  *
  *     representation of calibration curves.    *
  *                                              *
  ************************************************ å

unit HTGraph;

interface
äøn+å
uses
  dos,
  TPCrt,
  graph,
  Commo,
  TFEdit,
  HTCal;

var GraphMode,GraphDriver,MaxX,MaxY: integer;

procedure InitGraphics;
ä Initiates graphics on current drive å

procedure TermPlot;
ä Complete procedure for plotting and changing a calibration curve å

procedure HeatCtrl;
ä Complete procedure for plotting and storing a temp measurment å
ä-------------------------------------------------------------------------å

implementation


procedure InitGraphics;
var Errorcode : integer;
begin
  GraphDriver:=Detect;
  InitGraph(GraphDriver,GraphMode,'ÖtpÖ');
  Errorcode:=GraphResult;
  if Errorcode<>grOK then
  begin
    writeln('Graphics error: ',Grapherrormsg(Errorcode));
    writeln('Program Aborted');
    WaitKey;        ä Commo å
    RestoreCRTMode;
    Exit;
  end;
  SetTextJustify(CenterText,CenterText);
  SetTextStyle(DefaultFont,0,1);
  MaxX:=GetMaxX;
  MaxY:=GetMaxY;
  RestoreCRTMode;
  TextColor(white);
end;
ä End InitGraphics å


procedure TermPlot;
const step=2;
var x,y,x1,y1,k,j : integer;
    exist,OK,OK2,drop,found,quit,plot : boolean;
    f : termfile;
    filename,termID : string;
    place : word;
    T,Hold : termarray;
    term : termtype;
begin
  writeln('In which file is the termistor to plot ?');
  write('Filename: ');
  ReadFile(f,filename,found);       ä TFEdit å
  if not found then
    NoFile                          ä TFEdit å
  else
    begin
      writeln('Which termistor do you want to plot ?');
      write('Termistor ID-PS : ');
      readln(termID);
      SeekTerm(f,termID,exist,place);    ä TFEdit å
      if not exist then
        begin
          writeln('Termistor ',termID,' did not exist in ',filename,'.');
          writeln('Press any Key to return to menu.');
          WaitKey;                       ä Commo å
        end
      else
          GetData(f,place,T);            ä TFEdit å
      Close(f);
      if exist then
        repeat
          plot:=false;
          InitGraphics;
          SetGraphMode(GraphMode);
          ClearViewPort;
          OutText('Press NumLock and use arrows to move dot cursor and remove do
          OutTextXY(0,12,'Press Ä5Å to enter a dot on the screen. This removes t
          OutTextXY(0,25,'previous dot from the screen but does not alter the cu
          OutTextXY(0,38,'unless it is stored later on.');
          OutTextXY(0,51,'Press ÄEÅ to end plotting.');
          for k:=1 to range do
            begin
              x:=2*k+50;
              y:=round(-0.083*TÄkÅ+383);
              PutPixel(x,y,1);
            end;
          x:=round(MaxX/2);
          y:=round(MaxY/2);
          OK:=false;
          repeat
            OK2:=true;
            drop:=false;
            if GetPixel(x,y)<>0 then
              drop:=true;
            PutPixel(x,y,1);
            x1:=x;
            y1:=y;
            case ReadKey of
              '2' : y:=y+1;
              '4' : x:=x-step;
              '8' : y:=y-1;
              '6' : x:=x+step;
              '5' : if (x mod step)=0 then
                      begin
                        drop:=true;
                        k:=round((x-50)/2);
                        TÄkÅ:=round((y-383)/-0.083);
                        for j:=70 to MaxY do
                          PutPixel(x,j,0);
                        PutPixel(x,y,1);
                        x:=x+step;
                      end;
              'E','e' : OK:=true;
            end;
            if not drop then
              PutPixel(x1,y1,0);
          until OK ;

          RestoreCRTMode;
```

```
          writeln('Do you want to store the modified calibration curve? ÄY/*Å');
          writeln('The old calibration curve will be stored as old',termID,'.');
          case ReadKey of
            'Y','y' : repeat
                      quit:=true;
                      CheckData(T,OK);
                      if OK then
                        begin
                          Reset(f);
                          GetData(f,place,Hold);
                          RemoveItem(f,place);
                          term.data:=T;
                          term.ID:=termID;
                          Seek(f,FileSize(f));
                          write(f,term);      ä Write new curve to file å
                          term.data:=Hold;
                          term.ID:=ConCat('old',termID);
                          SeekTerm(f,term.ID,exist,place);
                          if exist then       ä Avoid double old curves å
                            RemoveItem(f,place);
                          Seek(f,FileSize(f));
                          write(f,term);      ä Write old curve to file å
                          Close(f);
                        end
                      else
                        begin
                          writeln('ÄMÅ to correct by manipulating.');
                          writeln('ÄPÅ to correct by plotting.');
                          writeln('ÄEscÅ to ignore new curve and return to menu
                          case ReadKey of
                            'M','m': begin
                                       Manipulate(T);
                                       quit:=false;
                                     end;
                            'P','p': plot:=true;
                            #¤1B   :;      ä ÄEscÅ Option å
                          end;
                        end;
                    until quit;
        end;
      until not plot;
    end;
end;
ä End TermPlot å

procedure DrawLine(time1,T1,time2,T2,timefactor:real;mintemp:integer);

var x1,x2,y1,y2:integer;

begin
  x1:=(round(0.1*MaxX)+round(0.85*timefactor*time1*MaxX));
  x2:=(round(0.1*MaxX)+round(0.85*timefactor*time2*MaxX));
  y1:=(round(0.9*MaxY)+round((mintemp-T1)*0.025*MaxY));
  y2:=(round(0.9*MaxY)+round((mintemp-T2)*0.025*MaxY));
  Line(x1,y1,x2,y2);
end;

procedure HeatCtrl;
var f : termfile;
    savef : text;
    quit,started,save,ignore,exist,found,inrange: boolean;
    timeint,timerange,maxtime,time_elapsed:integer;
    timefactor:real;
    temp1,temp2 : array Ä0..2Å of real;
    k,n,MinTemp : integer;
    chr : char;
    termID,filename,ak,outstr : string;
    hour1,hour2,minute1,minute2,second1,second2,hund1,hund2: word;
    place,year,month,day,dow: word;
    T : arrayÄ0..2Å of termarray;

begin
  writeln('In which file is calibration data?');
  ReadFile(f,filename,found);        ä TFEdit å
  if not found then
    NoFile                           ä TFEdit å
  else
    begin
     for k:=1 to 3 do
      begin
       str(k,ak);
       termID := ConCat(ak,ConCat( '-',ak));
       SeekTerm(f,termID,exist,place); ä TFEdit å
       HiddenCursor;
       if not exist then
        begin
         writeln('Termistor ',termID,' did not exist in ',filename);
         writeln('Press any key to return to menu');
         WaitKey;                    ä Commo å
         Exit;
        end
       else
         GetData(f,place,TÄk-1Å);         ä TFEdit å
      end;

     writeln('How long interval between measurements ? ( seconds )');
     repeat
       writeln('Time interval must be an integer between 1 and 600.');
       write('Time interval: ');
      ä¤I-å              ä Turn off automatic input error checking å
       readln(timeint);
      ä¤I+å              ä Turn on automatic input error checking å
     until (IOResult=0) and (timeint<=600) and (timeint>=1);

     writeln('For how long do you want to measure ? ( minutes )');
     repeat
       writeln('Duration must be an integer between 1 and 300.');
       write('Duration: ');
      ä¤I-å              ä Turn off automatic input error checking å
       readln(timerange);
      ä¤I+å              ä Turn on automatic input error checking å
     until (IOResult=0) and (timerange<=300) and (timerange>=1);

     writeln('Do you wish to save the measurment in a file? ÄY/*Å');
     save:=false;
     case ReadKey of
       'Y','y' : save:=true;
     end;

    if save then
     begin
      write('Filename (*.HTC) :');
      filename:='';
      repeat
       ignore:=false;
       found:=false;
       while not eoln do
        begin
         read(chr);
         if chr='.' then
          ignore:=true;
         if not ignore then
          filename:=ConCat(filename,chr);
        end;
       readln;
      until filename<>'';                ä Must have a filename å
      writeln;
      filename:=ConCat(filename,'.htc');
      Assign(savef,Concat(DestDir,filename));
     ä¤I-å      ä Turn off automatic input/output error checking å
```

```
 Append(savef);
ä¤I+å      ä Turn on automatic input/output error checking å
 if IOResult<>0 then
  Rewrite(savef);
end;

InitGraphics;
SetGraphMode(GraphMode);
ClearViewPort;
MinTemp:= MaxTemp - round(range/10);
timefactor:=timeint/(timerange*60);
maxtime:=round(1/timefactor);
OutText('HEATER ON : ÄHÅ');
OutTextXY(0,12,'HEATER OFF : ÄOÅ');
OutTextXY(0,25,'QUIT : ÄEscÅ');
OutTextXY(0,38,'START MEASUREMENT : ÄSpaceÅ');
OutTextXY(25,round(0.22*MaxY),'Temp ÄCÅ');
OutTextXY(round(MaxX*0.85),round(MaxY*0.92),'Time ÄminÅ');
OutTextXY(MaxX-200,round(0.1*MaxY),'Heater:');
OutTextXY(MaxX-140,round(0.1*MaxY),'OFF');
OutTextXY(round(MaxX*0.4),0,'PORT     TEMP');
OutTextXY(round(MaxX*0.4),12,' 1');
OutTextXY(round(MaxX*0.4),25,' 2');
OutTextXY(round(MaxX*0.4),38,' 3');
DrawLine(0,MinTemp,maxtime,MinTemp,timefactor,MinTemp); (* Drar x-axeln; ti
DrawLine(0,MinTemp,0,MaxTemp,timefactor,MinTemp);  (* Drar y-axeln; tempera
for k:=0 to timerange do
 DrawLine(round(60*k/timeint),(MinTemp-0.25),
          round(60*k/timeint),MinTemp,timefactor,MinTemp); (* Skalar x-axel
for k:=0 to round(MaxTemp-MinTemp) do
 if ((MinTemp+k) mod 5 ) = 0 then
  begin
   DrawLine(-0.02*maxtime,(MinTemp+k),0,(MinTemp+k),timefactor,MinTemp);  (
   str((MinTemp+k),outstr);
   OutTextXY(round(0.05*MaxX),round(MaxY*(0.9-k*0.025))-4,outstr);
  end
else
  DrawLine(-0.01*maxtime,(MinTemp+k),0,(MinTemp+k),timefactor,MinTemp);  (*
k:=0;
quit:=false;
started:=false;
repeat
 if KeyPressed then
   case ReadKey of
     #¤1B   : quit:=true;
     'H','h': begin
                HighOut;
                SetViewPort(MaxX-140,round(0.1*MaxY),
                            MaxX,round(0.1*MaxY+15),true);
                ClearViewPort;
                SetViewPort(0,0,MaxX,MaxY,true);
                OutTextXY(MaxX-140,round(0.1*MaxY),'ON ');
               end;
     'O','o': begin
                LowOut;
                SetViewPort(MaxX-140,round(0.1*MaxY),
                            MaxX,round(0.1*MaxY+15),true);
                ClearViewPort;
                SetViewPort(0,0,MaxX,MaxY,true);
                OutTextXY(MaxX-140,round(0.1*MaxY),'OFF');
               end;
     #¤20   : if not started then
               begin
                for n:=0 to 2 do
                 GetTemp(TÄnÅ,n+1,temp1ÄnÅ,inrange);
                GetTime(hour1,minute1,second1,hund1);
                SetViewPort(0,38,round(MaxX*0.3),50,true);
                ClearViewPort;
                SetViewPort(0,0,MaxX,MaxY,true);
                if save then
                begin
                 GetDate(year,month,day,dow);
                 writeln(savef,'Temperature measurement   ',day:2,'/',
                         month:2,'  ',year:4);
                 writeln(savef,'  TIME     PORT 1    PORT 2    PORT 3');
                 writeln(savef,hour1:2,':',minute1:2,':',second1:2,
                         '     ',temp1Ä0Å:4:1,'      ',temp1Ä1Å:4:1,
                         '     ',temp1Ä2Å:4:1);
                end;
                started:=true;
               end;
  end;
 if started then
  begin
   GetTime(hour2,minute2,second2,hund2);
   time_elapsed:=(hour2-hour1)*3600+(minute2-minute1)*60+(second2-second1);
   if time_elapsed >= timeint then
    begin
     if save then
       write(savef,hour2:2,':',minute2:2,':',second2:2,'    ');
     for n:=0 to 2 do
      begin
       GetTemp(TÄnÅ,n+1,temp2ÄnÅ,inrange);
       if inrange then
        begin
         DrawLine(k,temp1ÄnÅ,k+1,temp2ÄnÅ,timefactor,MinTemp);
         SetViewPort(round(MaxX*0.5),round(12*n+13),
                     round(MaxX*0.7),round(12*n+25),true);
         ClearViewPort;
         str(temp2ÄnÅ:4:1,outstr);
         OutText(outstr);
         SetViewPort(0,0,MaxX,MaxY,true);
         if save then
           write(savef,temp2ÄnÅ:4:1,'      ');
        end
       else
        begin
         SetViewPort(round(MaxX*0.5),round(12*n+13),
                     round(MaxX*0.7),round(12*n+25),true);
         ClearViewPort;
         OutText('Out of range');
         SetViewPort(0,0,MaxX,MaxY,true);
         if save then
           write(savef,'OofR','      ');
        end;
       temp1ÄnÅ:=temp2ÄnÅ;
      end;
     hour1:=hour2;
     minute1:=minute2;
     second1:=second2;
     hund1:=hund2;
     k:=k+1;
     if save then
       writeln(savef);
    end;
  end;
until (k=maxtime) or quit;

LowOut;
SetViewPort(MaxX-140,round(0.1*MaxY),
            MaxX,round(0.1*MaxY+15),true);
ClearViewPort;
SetViewPort(0,0,MaxX,MaxY,true);
OutTextXY(MaxX-140,round(0.1*MaxY),'OFF');
OutTextXY(0,38,'Press any key to return to menu');
WaitKey;

RestoreCRTMode;
NormalCursor;
Close(f);
```

```
421:       if save then
422:         Close(savef);
423:       Window(1,1,80,25);
424:     end;
425: end;
426: ä End HeatCtrl å
427:
428:
429:
430: end.
431: ä End HTGraph å
```

```
ä ****************************************************
  *                                                  *
  *                  UNIT HTCal                      *
  *  This unit contains procedures for calibration   *
  *    of termistors and ports.It also contains      *
  *  general procedures for temperature measurements.*
  *                                                  *
  **************************************************** å

unit HTCal;

interface

uses TPCrt,
     Commo,
     TFEdit;

procedure AutoRead;
ä Complete procedure for monitoring the ports on the ADDA card å

procedure GetTemp(T:termarray;port:word;var temp:real;var inrange:boolean);
ä General procedure for converting a port value to a temperature å

procedure StoreCal(T:termarray;portnr:string);
ä General procedure for storing calibration data å

procedure ManuCal;
ä Complete procedure for manual calibration of a termistor å

procedure AutoCal;
ä Complete procedure for automatic calibration of a termistor å

ä------------------------------------------------------------------------------å

implementation

procedure AutoRead;
var value : integer;
    port: word;
    quit : boolean;
begin
 writeln('Which port do you want to monitor ? ');
 ReadPortNr(port);                              ä Commo å
 ClrScr;
 writeln('Press ÄEscÅ to return to menu.');writeln;
 writeln('Port ',port,':');
 quit:=false;
 Window(10,3,80,25);
 HiddenCursor;
 repeat
   ClrEol;
   value:=ReadPort(port);                       ä Commo å
   writeln(value);
   Delay(1000);
   GotoXY(1,1);
   if Keypressed then if ReadKey=#¤1B then quit:=true;
 until quit;
 Window(1,1,80,25);
 NormalCursor;
end;
ä End AutoRead å

procedure GetTemp(T:termarray;port:word;var temp:real;var inrange:boolean);
var  value,k : integer;
     quit : boolean;
begin
  value:=ReadPort(port);                ä Commo å
  inrange:=true;
  k:=0;
  quit:=false;
  repeat
    if value>TÄkÅ then
      if k=range then
        begin inrange:=false; quit:=true; end    ä Below temp interval å
      else
        k:=k+1                                   ä Continue search å
    else
      if k=0 then
        begin inrange:=false; quit:=true; end    ä Above temp interval å
      else
        begin
          if ((value-TÄk-1Å)/(TÄkÅ-TÄk-1Å)) < 0.5 then
            k:=k-1;                              ä Round to nearest value å
          quit:=true;
        end;
  until quit;
  temp:=(maxtemp-k/10);
end;
ä End GetTemp å


procedure StoreCal(T:termarray;portnr:string);
var OK,OK2,exist,found : boolean;
    place : word;
    f : termfile;
    term : termtype;
    filename,termID : string;

begin
 ClrScr;
 repeat
   Scroll(T);                                                  ä TFEdit å
   writeln('Do you want to manipulate the data? ÄY/*Å');
   case ReadKey of
     'Y','y' :  Manipulate(T);                                 ä TFEdit å
   end;
   CheckData(T,OK);                                            ä TFEdit å
 until OK;
 ClrScr;
ä Storing å
 repeat
   writeln('Enter the name of the file in which you ');
   writeln('want to store the calibration data.( *.CAL )');
   ReadFile(f,filename,found);                                       ä TFEdit å
   OK:=true;
   if not found then
     begin
       writeln('File did not exist. Do you want to create it ? ÄY/*Å');
       case ReadKey of
         'Y','y' : Create(f);                                  ä TFEdit å
       else
         OK:=false;
       end;
     end;
   if OK then    ä File to write to exists å
      repeat
       OK2:=true;
       writeln('Enter ID number of termistor and remember to mark it');
       readln(termID);
       termID:=ConCat(termID,ConCat('-',portnr));  ä Make ID in ID-PS form å
       SeekTerm(f,termID,exist,place);                          ä TFEdit å
       if exist then
         begin
           writeln('That termistor already exist in ',filename,'. Do you want');
           writeln('to replace the existing data with the new ? ÄY/*Å');
           case ReadKey of
             'Y','y' : begin
                         RemoveItem(f,place);                   ä TFEdit å
                         term.ID:=termID;
                         term.data:=T;
```

```
                        Seek(f,FileSize(f));
                        write(f,term);
                      end;
            else
              OK2:=false;
            end;
          end
        else     ä Termistor did not exist in file already å
          begin
            term.ID:=termID;
            term.data:=T;
            write(f,term);
          end;
      until OK2;
  until OK;
  Close(f);
  writeln('Calibration stored. Press any key to return to menu.');
  WaitKey;                                          ä Commo å
  Window(1,1,80,25);
  ClrScr;
end;
ä End StoreCal å

procedure ManuCal;
var k : integer;
    OK : boolean;
    T : termarray;
    filename,portnr : string;
    port : word;

begin
 repeat
  OK:=true;
  writeln('On which port do you want to calibrate the termistor?');
  writeln('Available ports: 1 2 3');
  ReadPortNr(port);                  ä Commo å
  case port of
    1..3 : str(port,portnr);
  else
    begin
     writeln('Port ',port,' does not exist.');
     OK:=false;
    end;
  end;
 until OK;
 ClrScr;
 writeln('Plug in the termistor to be calibrated on port ',portnr,'.');
 writeln('Put the termistor in water that is heated to above ',Maxtemp,' deg C.'
 writeln('Place a thermometer near the termistor in the water.');
 writeln('Let the water cool off at any rate you wish to less than ',(Maxtemp-ro
 writeln;
 writeln('* ÄSpaceÅ  :  Register temperature displayed on the screen');
 writeln('* ÄIÅ      :  Ignore last value');
 writeln('* ÄHÅ      :  Turn on heater ( WARNING! Must be in water! ).');
 writeln('* ÄOÅ      :  Turn off heater ');
 writeln('* ÄEscÅ    :  Abort calibration and return to main menu ');
 Window(26,15,55,18);
 GotoXY(1,1);
 write(#¤C9);for k:=1 to 27 do write(#¤CD);writeln(#¤BB);
 writeln(#¤BA,'                          ',#¤BA);
 write(#¤C8);for k:=1 to 27 do write(#¤CD);writeln(#¤BC);
 Window(28,16,53,17);
 GotoXY(1,1);
 HiddenCursor;
 write(MaxTemp,' deg C ');
 for k:=0 to range do           ä Start calibration å
  begin
   GotoXY(1,1);
   repeat
    OK:=true;
    case ReadKey of
      #¤20    : begin                                   ä ÄSpaceÅ å
                  TÄkÅ:=ReadPort(port);                 ä Commo å
                  if k > 0 then
                   if TÄkÅ < TÄk-1Å then
                    TÄkÅ:=TÄk-1Å;
                end;
      'H','h' : begin HighOut; k:=k-1; end;             ä Commo å
      'O','o' : begin LowOut;  k:=k-1; end;             ä Commo å
      'I','i' : if k>0 then k:=k-2 else k:=k-1;
      #¤1B    : begin Window(1,1,80,25); NormalCursor; Exit; end;  ä ÄEscÅ å
    else
      OK:=false
    end;
   until OK;
   ClrEol;
   write(' -(k+1)/10 + Maxtemp:3:1,' deg C  Port ',portnr,': ',TÄkÅ);
  end;
LowOut;    ä Turn off heater å                          ä Commo å
Window(1,1,80,25);
NormalCursor;
StoreCal(T,portnr);
end;
ä End ManuCal å

procedure AutoCal;
var k,value,v1,v2,v3 : integer;
    OK,exist,inrange,above,found : boolean;
    portnr1,portnr2 : string;
    temp : real;
    port1,port2,place : word;
    filename,termID : string;
    f : termfile;
    Tk,Tuk : termarray;

begin
  writeln('Automatic calibration requires a known termistor.');
  repeat
    OK:=true;
    writeln('On which port do you want to calibrate the termistor? (1, 2 or 3)')
    ReadPortNr(port1);                              ä Commo å
    case port1 of
      1..3  : str(port1,portnr1);
    else
      begin
       writeln('Port ',port1,' does not exist.');
       OK:=false;
      end;
    end;
  until OK;
  repeat
    OK:=true;
    writeln('On which port do you intend to plug in the known termistor ?');
    ReadPortNr(port2);                              ä Commo å
    if port1=port2 then
      begin
        writeln('You cannot use the same port as before.');
        OK:=false;
      end
    else
      case port2 of
        1..3    : str(port2,portnr2);
      else
        begin
         writeln('Port ',port2,' does not exist.');
         OK:=false;
        end;
      end;
```

```
281:   until OK;  ä Both ports exist å
282:   writeln('Enter the name of the file where the known termistor is to be found.'
283:   write('Filename: ');
284:   ReadFile(f,filename,found);                                    ä TFEdit å
285:   exist:=false;
286:   if not found then
287:     NoFile                                                       ä TFEdit å
288:   else
289:     begin
290:       write('Termistor ID of known termistor: ');
291:       readln(termID);
292:       termID:=ConCat(ConCat(termID,'-'),portnr2);
293:       SeekTerm(f,termID,exist,place);                            ä TFEdit å
294:       if exist then
295:         GetData(f,place,Tk)                                      ä TFEdit å
296:       else
297:         NoTerm;                                                  ä TFEdit å
298:       Close(f);
299:     end;
300:   if exist then    ä Ready for calibration å
301:     begin
302:       ClrScr;
303:       writeln('Plug in known termistor (',termID,') on port ',portnr2,' and the
304:       writeln('calibrated on port ',portnr1,'.Heat water to above ',MaxTemp,' de
305:       writeln('and place the termistors close together in the water. Let it cool
306:       writeln('to less than ',round(MaxTemp-range/10),' deg C while being stirre
307:       writeln;
308:       writeln('* ÄEscÅ    :  Abort calibration and return to main menu ');
309:       writeln('* ÄHÅ      :  Turn on heater ( WARNING! Must be in water! ).');
310:       writeln('* ÄOÅ      :  Turn off heater ');
311:       writeln('* ÄSpaceÅ  :  Start calibration.');
312:       HiddenCursor;
313:       repeat
314:         OK:=false;
315:         case ReadKey of
316:          #¤20    : OK:=true;                                       ä ÄSpaceÅ å
317:          'H','h' : HighOut;                        ä Commo å
318:          'O','o' : LowOut;                         ä Commo å
319:          #¤1B    : begin Window(1,1,80,25); NormalCursor; Exit; end; ä ÄEscÅ å
320:         else; end;
321:       until OK;
322:       LowOut;                                                    ä Commo å
323:       ClrScr;
324:       writeln('Press ÄEscÅ to abort calibration.');
325:       Window(15,15,62,20);
326:       GotoXY(1,1);
327:       write(#¤C9);for k:=0 to 44 do write(#¤CD);writeln(#¤BB);
328:       writeln(#¤BA,'                                              ',#¤BA);
329:       write(#¤C8);for k:=0 to 44 do write(#¤CD);writeln(#¤BC);
330:       Window(17,16,59,18);
331:       GotoXY(1,1);
332:
333:     ä Calibration å
334:       repeat
335:         GetTemp(Tk,port2,temp,inrange);
336:         v1:=ReadPort(port1);              ä Commo å
337:         write('Not in interval yet. Temp: ',temp:1:1,' deg C.');
338:         GotoXY(1,1);
339:         if KeyPressed then case ReadKey of
340:            #¤1B    : begin Window(1,1,80,25); NormalCursor; Exit; end; ä ÄEscÅ å
341:                      end;
342:       until inrange;    ä Upper limit of temp interval reached å
343:       v2:=v1;
344:       for k:=0 to (range-1) do
345:         begin
346:          above:=false;
347:          repeat
348:            OK:=false;
349:            if Keypressed then case ReadKey of
350:              #¤1B    : begin Window(1,1,80,25); NormalCursor; Exit; end; ä ÄEscÅ
351:                                end;
352:           value:=ReadPort(port2);      ä Commo å
353:           if value > TkÄk+1Å then       ä Next temp reached å
354:             begin
355:               v3:=ReadPort(port1);     ä Commo å
356:               TukÄkÅ:=round((v1+v2)/2);  ä Enter calibration point å
357:               if ( k > 0 ) and ( TukÄkÅ < TukÄk-1Å ) then
358:                 TukÄkÅ:=TukÄk-1Å;
359:               v1:=v3;
360:               v2:=v1;
361:               OK:=true;
362:             end;
363:           ClrEol;
364:           write(-k/10+MaxTemp:3:1,' deg C  Port ',portnr2,'Ä',TkÄk+1Å,'Å:');
365:           write(value,' Port ',portnr1,':',ReadPort(port1));   ä Commo å
366:           GotoXY(1,1);
367:           if value < TkÄkÅ then         ä Temp rises above current temp å
368:             above:=true;
369:           if above and (value > TkÄkÅ) then   ä Temp drops below current temp a
370:             begin
371:               v2:=ReadPort(port1);         ä Commo å
372:               above:=false;
373:             end;
374:         until OK;
375:       end;
376:       TukÄrangeÅ:=v1;
377:       LowOut;            ä Make sure that heater is off å     ä Commo å
378:       Window(1,1,80,25);
379:       NormalCursor;
380:       StoreCal(Tuk,portnr1);
381:     end;
382:   end;
383: ä End AutoCal å
384:
385: end.
386: ä End HTCal å
```

```
 1: program termoreg;
 2: ägn+å
 3: uses
 4:   dos,crt,graph;
 5: var GraphMode,GraphDriver,MaxX,MaxY,para : integer;
 6:     I,K,T0,Tb,bpef,opd,value : double;
 7:     wish : stringÄ1Å;
 8:     slut : boolean;
 9:
10: procedure Initgraphics;
11: var Errorcode: integer;
12: begin
13:   GraphDriver:= Detect;
14:   InitGraph(GraphDriver,GraphMode,'ÖtpÖ');
15:   Errorcode:= GraphResult;
16:   if Errorcode<>grOK then
17:   begin
18:     writeln('Graphics error: ',Grapherrormsg(Errorcode));
19:     writeln('Program Aborted');
20:     Halt(1);
21:   end;
22:   SetTextJustify(CenterText,CenterText);
23:   SetTextStyle(DefaultFont,0,1);
24:   MaxX:= GetMaxX;
25:   MaxY:= GetMaxY;
26:   RestoreCRTMode;
27:   TextColor(white);
28: end;
29:
30: procedure DrawLine(mm1,T1,mm2,T2:double);
31: var x1,x2,y1,y2:integer;
32:
33: begin
34:   x1:=(round(0.2*MaxX)+round(0.05*mm1*MaxX));
35:   x2:=(round(0.2*MaxX)+round(0.05*mm2*MaxX));
36:   y1:=(round(0.9*MaxY)+round((30-T1)*0.0364*MaxY));
37:   y2:=(round(0.9*MaxY)+round((30-T2)*0.0364*MaxY));
38:   Line(x1,y1,x2,y2);
39: end;
40:
41:
42: procedure PlotTemp(heatint,opd,surftemp,bloodtemp,heatcond,bpef:double);
43: const maxdepth=12;      (* Djup i mm, temperaturer i C *)
44:       mintemp=30;
45:       maxtemp=52;
46:       treattemp=43;
47:       termdepth=5;
48: var  ON : char;
49:      k,n: integer;
50:      gamma,sigma,A,temp1,temp2:double;
51:
52: begin
53:   sigma:=1/opd;
54:   gamma:=bpef/heatcond;
55:   A:=heatint*sigma/(heatcond*(sqr(sigma)-gamma));
56:   SetGraphMode(GraphMode);
57:   DrawLine(0,mintemp,maxdepth,mintemp); (* Drar x-axeln; djupet i mm *)
58:   DrawLine(0,mintemp,0,maxtemp);  (* Drar y-axeln; temperatur i C *)
59:   for k:=1 to maxdepth do
60:    DrawLine(k,(mintemp-0.25),k,mintemp); (* Skalar x-axeln *)
61:   for k:=1 to round(maxtemp-mintemp) do
62:    DrawLine(-0.1,(mintemp+k),0,(mintemp+k));  (* Skalar y-axeln *)
63:   DrawLine(-0.2,treattemp,0,treattemp);  (* Markerar intressant temperatur *)
64:   DrawLine(termdepth,(mintemp-0.5),termdepth,mintemp); (* Mark. termistor *)
65:   for n:=0 to 20 do
66:     begin
67:       temp1:=surftemp+n;
68:       for k:=1 to 12 do
69:       begin
70:         temp2:=( (surftemp+n) - bloodtemp + A )*exp(-sqrt(gamma)*k) - A*exp(-sig
71:         DrawLine((k-1),temp1,k,temp2);
72:         temp1:=temp2;
73:       end;
74:     end;
75:   ON:=ReadKey;
76:   RestoreCRTMode;
77: end;
78:
79: (****** MAIN PROGRAM ******)
80: begin
81:   ClrScr;
82:   Initgraphics;
83:   writeln('Heat Intensity ÄmW/mmÜ2Å (1) ?');
84:   readln(I);
85:   writeln('Optical penetration depth ÄmmÅ (2) ?');
86:   readln(opd);
87:   writeln('Heat conductance ÄmW/mm CÅ (3) ?');
88:   readln(K);
89:   writeln('Blood perfusion exchange factor ÄmW/mmÜ3 CÅ (4) ?');
90:   readln(bpef);
91:   writeln('Surface temperature Ä CÅ (5) ?');
92:   readln(T0);
93:   writeln('Blood temperature Ä CÅ (6) ?');
94:   readln(Tb);
95:   repeat
96:    PlotTemp(I,opd,T0,Tb,K,bpef);
97:    slut:=true;
98:    repeat
99:       writeln('Change Parameter ? (y/n)');
100:      readln(wish);
101:      if wish='y' then
102:      begin
103:        slut:=false;
104:        writeln('(1)Heatint: ',I:6:4,' mW/mmÜ2');
105:        writeln('(2)OPD: ',opd:6:4,' mm');
106:        writeln('(3)Heat conductance: ',K:6:4,' mW/mm C');
107:        writeln('(4)Blood perf EF: ',bpef:6:4,' mW/mmÜ3 C');
108:        writeln('(5)Surface temp: ',T0:6:4,'  C');
109:        writeln('(6)Blood temp: ',Tb:6:4,'  C');
110:        writeln('Which ? (N:o 1,2,3,4,5 or 6)');
111:        readln(para);
112:        writeln('New Value ?' );
113:        readln(value);
114:        case para of
115:         1:I:= value;
116:         2:opd:= value;
117:         3:K:= value;
118:         4:bpef:= value;
119:         5:T0:= value;
120:         6:Tb:= value;
121:        else
122:         Writeln('No Such Parameter.');
123:        end;
124:      end;
125:    until (wish='n');
126:
127:   until slut;
128: end.
```