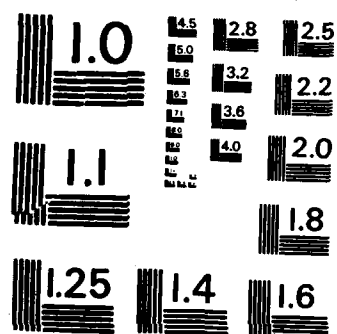


UNCLASSIFIED

USACRREL (US ARMY COLD REGIONS RESEARCH & ENGINEERING
LABORATORY) PRECISE (U) COLD REGIONS RESEARCH AND
ENGINEERING LAB HANOVER NH G M TRACHIER ET AL DEC 85
CRREL-SR-85-26 F/G 14/2

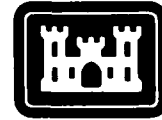
ML



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Special Report 85-26

December 1985



12

**US Army Corps
of Engineers**

Cold Regions Research &
Engineering Laboratory

USACRREL precise thermistor meter

G.M. Trachier, J.S. Morse and S.F. Daly

AD-A166 470

DTIC
ELECTE
APR 10 1986
S D

DTIC FILE COPY

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Special Report 85-26	2. GOVT ACCESSION NO. ADA 166 470	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) USACRREL PRECISE THERMISTOR METER		5. TYPE OF REPORT & PERIOD COVERED
7. AUTHOR(s) G.M. Trachier, J.S. Morse and S.F. Daly		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. Army Cold Regions Research and Engineering Laboratory Hanover, New Hampshire 03755-1290		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DA Project 4A161101A91D
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE December 1985
		13. NUMBER OF PAGES 39
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution is unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Frazil ice Ice Instrument fabrication Water temperature measurement		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) To facilitate the study of frazil ice in the field, a highly accurate, portable water temperature meter was required. The USACRREL Precise Thermistor Meter was designed and built to meet this need. The meter is rugged, battery-operated, waterproof, and able to operate over a wide range of ambient temperatures. A unique feature of this instrument is the use of software to compensate for temperature-dependent variation in the analog electronics. The circuitry consists of an analog printed circuit board and a low power micro-computer. The resistance of a calibrated thermistor is determined and its temperature calculated using the Steinhart-Hart equation. The accuracy of the meter was determined		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

20. Abstract (cont'd).

both theoretically and in cold room tests. The hardware and software used in the meter are described.

very good

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

PREFACE

This report was prepared by Gary M. Trachier, Electronics Technician, James S. Morse, Electronics Engineer, both of the Engineering and Measurement Services Branch, Technical Services Division, and Steven F. Daly, Research Hydraulic Engineer, Ice Engineering Research Branch, Experimental Engineering Division, U.S. Army Cold Regions Research and Engineering Laboratory. Funding was provided by DA Project 4A161101A91D, In-House Laboratory Independent Research.

Steven F. Daly determined the instrument's capabilities and requirements. James Morse conceived the idea of interfacing a thermistor and resistance-to-voltage converter to a microprocessor. Gary Trachier selected the modules and components, built the instrument, developed the software and tested the final product. The authors thank John Kalafut for his assistance in building the instrument, Robert Demars and David L'Heureux for their precision photography and assistance in making the printed circuit boards, Mark Hardenberg for editing this report and Matthew Pacillo for drafting the figures.

The contents of this report are not to be used for advertising or promotional purposes. Citation of brand names does not constitute an official endorsement or approval of the use of such commercial products.

Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



CONTENTS

	Page
Abstract.....	1
Preface.....	iii
Introduction.....	1
Instrument requirements.....	2
Instrument development.....	2
General.....	2
Hardware.....	4
Software.....	7
Instrument accuracy.....	10
Errors external to the instrument.....	10
Internal instrument errors.....	11
Instrument error analysis.....	12
Summary.....	16
Literature cited.....	17
Appendix A: Programs.....	19
Appendix B: Instructions for operation of thermistor meter Version 1.0.....	33

ILLUSTRATIONS

Figure

1. USACRREL precise thermistor meter.....	1
2. Instrument circuit block diagram.....	3
3. Hardware.....	4
4. Schematic of analog PCB circuit.....	5
5. Output units of onboard temperature sensor.....	7
6. Analog PCB offset temperature compensation.....	11
7. Calibration curves for A/D converter.....	13
8. Error analysis.....	16

TABLES

Table

1. Thermistor at 0°C with instrument temperature varied.....	13
2. Thermistor and instrument at the same temperature.....	14
3. Calibration on 9 and 10 January 1985.....	15

USACRREL PRECISE THERMISTOR METER

G.M. Trachier, J.S. Morse and S.F. Daly

INTRODUCTION

The study of ice in rivers, and particularly frazil ice, has been frustrated by the lack of an instrument capable of determining water temperature with sufficient accuracy and precision. For example, in the study of frazil ice produced in rivers and streams, it is well known that supercooling of the water during frazil production in the field rarely exceeds 0.03°C , and never exceeds 0.05°C . It is vitally important that temperatures of supercooling be measured accurately to within 0.01°C for us to increase our understanding of the fundamentals of frazil formation.

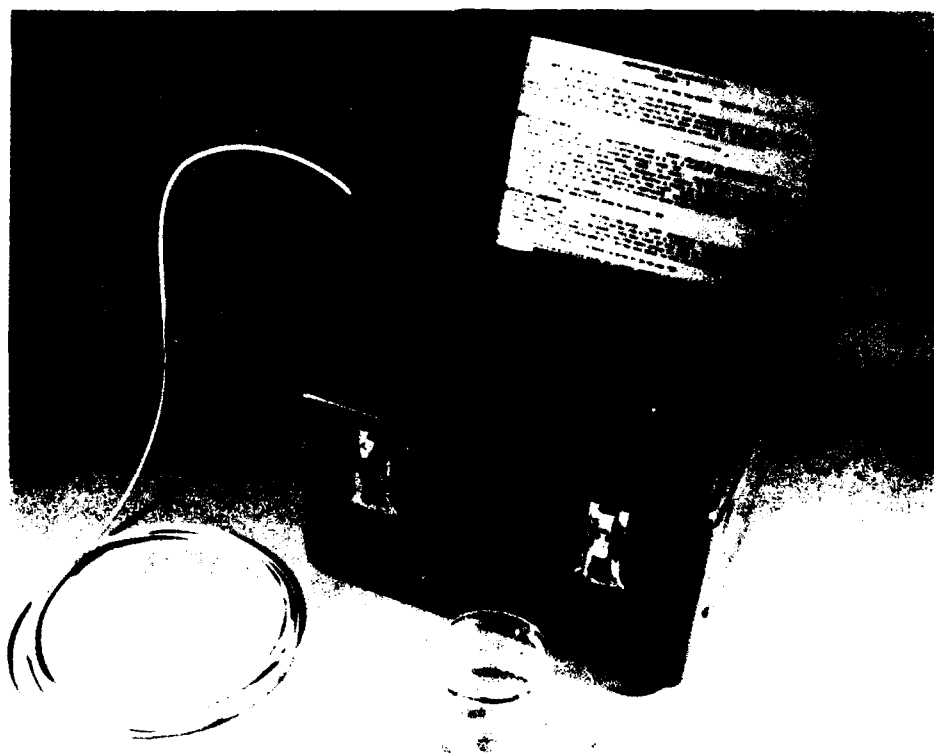


Figure 1. USACRREL precise thermistor meter.

Many other ice phenomena may also be critically dependent on small changes of supercooling. However, until now there has existed no temperature measuring device that is both portable and rugged enough to be used in the field and that has the precision and accuracy necessary.

This report describes an accurate, precise temperature measurement instrument recently developed at CRREL (Fig. 1). A unique feature of this instrument is the use of software to compensate for temperature-dependent variation in the instrument's circuitry. This greatly improves the accuracy of the instrument over a wide range of ambient temperatures.

INSTRUMENT REQUIREMENTS

The meter would be primarily used to measure water temperature under a variety of field conditions. It had to be built to meet the following requirements:

1. Accuracy -- the meter should be accurate to $0.01^{\circ}\text{C} \pm 0.025^{\circ}\text{C}$.
2. Operating conditions -- the meter should be able to operate under a variety of field conditions in temperatures ranging from -30° to 20°C . It must also be waterproof, buoyant, and rugged enough to withstand the rigors of field use.
3. Usability -- the meter should have a digital readout of temperature. It should be hand-held and convenient to use. To allow maximum portability, it should be battery operated.

INSTRUMENT DEVELOPMENT

General

Given the meter requirements outlined in the previous section, a survey was made of the commercially available digital thermometers; none that approached the required accuracy and resolution were suitable for field use. The majority required a 110-Vac power source and all would operate correctly under a very limited ambient temperature range. In addition, none were rugged enough or waterproof. Therefore, we decided to develop and construct a unit at CRREL.

The choice of temperature sensors was narrowed down to either a Platinum Resistance Temperature Device (PtRTD) or a calibrated thermistor. Both are widely used in industry and known for their long-term stability. We chose the thermistor. Thermistors have a much larger change in

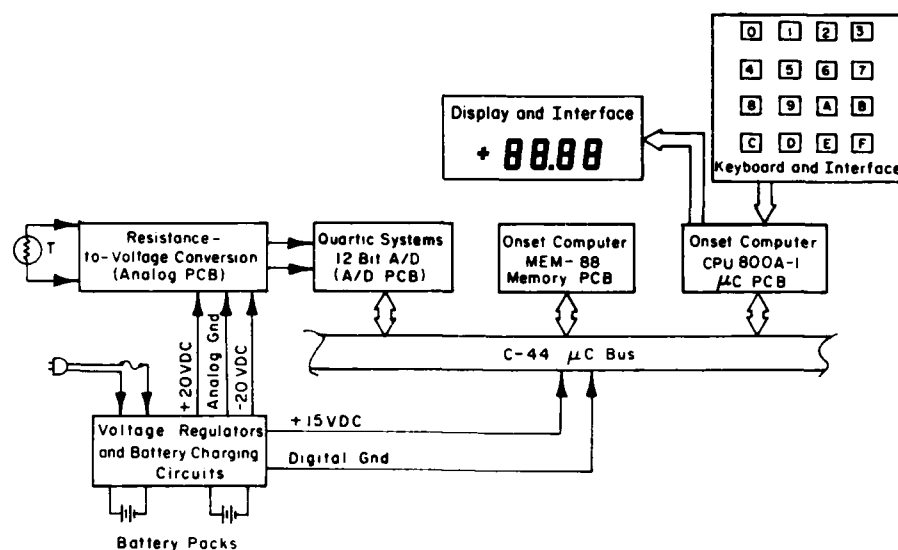


Figure 2. Instrument circuit block diagram.

resistance per unit temperature change than a PtRTD, allowing greater accuracy in measuring small changes in temperature.

There are two popular methods for precisely measuring resistance. One uses the Wheatstone bridge, where the unknown resistance is matched to a known resistance. The other uses a constant current that is passed through the thermistor while the voltage across it is measured. The resistance can then be calculated using Ohm's law. This is the principle behind a standard ohmmeter. These two methods are well proven, but inconvenient for determining temperature. The user must first determine the thermistor's resistance. He must then look up the resistance on a special table relating resistance to temperature. One of these tables must be on hand for each thermistor being used.

We decided to use a calibrated thermistor, constant current and an analog-to-digital (A/D) converter to measure the voltage (Fig. 2). This voltage would then be used to determine the thermistor's resistance. To avoid the inconvenience of conversion tables, we included a microcomputer in the instrument to convert the thermistor's resistance to a temperature reading. The calculated temperature is then displayed.

The known resistance of the thermistor can be used to determine its temperature by use of the Steinhart-Hart equation (Steinhart and Hart 1968, Yellow Springs Instruments Inc. 1971). The constants for the Steinhart-Hart equation would be entered by the user into memory, and remain there until changed.

The Steinhart-Hart equation, which describes the temperature of a thermistor as a function of its resistance, is

$$\frac{1}{T} = A + B * (\ln R) + C * (\ln R)^3 \quad (1)$$

where T = temperature (Kelvins)

R = thermistor resistance (ohms)

A,B,C = fitting constants (determined through the thermistor calibration).

Hardware

The electronics are housed in a case that is 7 in. wide, 11 in. long and 8-1/2 in. high, which can be hand carried. The case has been sealed thoroughly with silicone sealant on all through-holes, cutouts and weld seams. This includes sealing the joint between the top panel and the lower box, and around all mounting holes on the top panel. Note that the hinged cover does not make a waterproof seal when closed. The instrument's electronics consist of an analog Printed Circuit Board (PCB) that converts the thermistor's resistance to a voltage, and a low power microcomputer that converts the voltage to a temperature reading and displays it. The

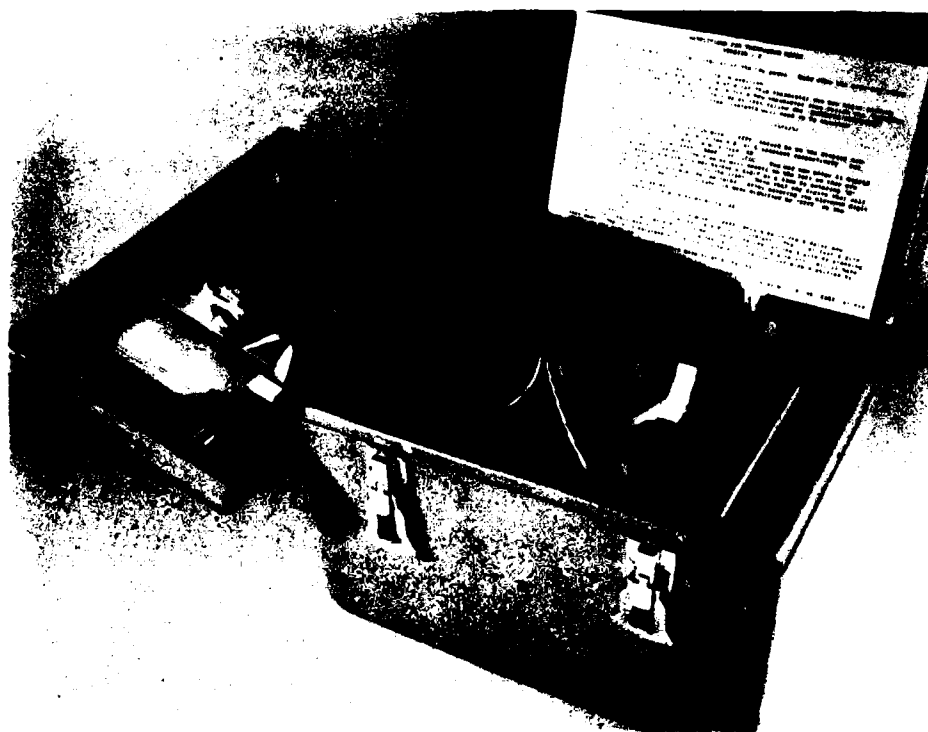


Figure 3. Hardware.

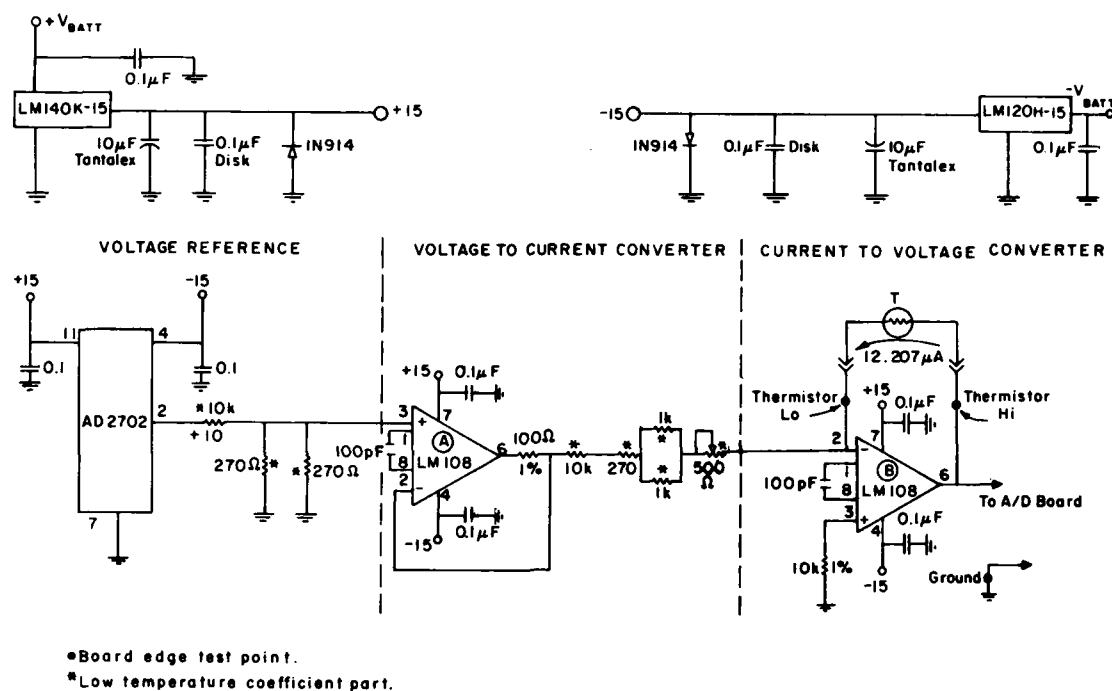


Figure 4. Schematic of analog PCB circuit (low temperature coefficient bypass capacitors used; a bypass capacitor is placed between the digital and analog grounds on the PCB).

microcomputer consists of a processor board, memory board and an A/D converter board. The microcomputer programs were created on a National Semiconductor Corporation Starplex II microprocessor development system. With this system, the microcomputer programs were written in a higher level language, compiled, interactively debugged and then "down loaded" to the microcomputer. Figure 3 shows the hardware.

Analog board

The schematic diagram of the analog PCB is shown in Figure 4. The analog PCB consists of the circuitry needed to convert the thermistor's resistance to a millivolt value and can be divided into two sections. The first portion of the circuit is a voltage-to-current converter. It is a standard "op-amp" (operational amplifier) circuit that converts the output of the precision voltage reference to a constant current source (Stout and Kaufman 1976). The AD2702UD precision voltage reference is the most stable one available without an internal heater. Its nominal output is 10.000 Vdc, with a low (5 ppm/°C) temperature coefficient. We wanted one without a heater to minimize power consumption from the battery pack. The LM108AHM

operational amplifier also has a very low power requirement and temperature coefficient ($1 \mu\text{V}/^\circ\text{C}$). All of the resistors in this part of the circuit have temperature coefficients of only $2 \text{ ppm}/^\circ\text{C}$.

The last part of the circuit is a current-to-voltage converter. The thermistor is placed in the op-amp's feedback loop. This assures that a constant current of $12.207 \mu\text{A}$ is passed through it. The op-amp's output voltage is then linearly proportional to the thermistor's resistance (Stout and Kaufman 1976).

Microcomputer system

Processor board. The processor board controls all Input/Output (I/O), corrects the voltage for temperature-dependent variation, determines the thermistor's resistance, and solves the Steinhart-Hart equation to determine the thermistor's temperature. It has a National Semiconductor Corporation (NSC800) microprocessor, 2 kilobytes of Erasable, Programmable, Read Only Memory (EPROM) and 2 kilobytes of Random Access Memory (RAM). It also contains the I/O ports that interface to the keyboard and Liquid Crystal Display (LCD). The machine cycle time of the microcomputer is $1 \mu\text{s}$.

Memory board. The program and thermistor constants are stored on the memory board. Seven 27C16 2-kilobyte by 8-bit EPROMs and one 52B13 2-kilobyte by 8-bit Electrically Alterable, Read Only Memory (EAROM) are mounted on this board. The program is stored in the former and the thermistor fitting constants in the latter. The EAROM requires a 10 ms write cycle, so a hardware time delay was designed to lengthen the machine write cycle from $1 \mu\text{s}$ to 10 ms.

A/D board. The A/D board converts the millivolt output from the analog PCB to digital. A 12-bit CMOS integrating converter does this. There are eight differential input channels that can be selected individually and a programmable gain amplifier. The six ranges of the programmable gain amplifier have full scale input voltages of 5.0, 2.5, 1.0, 0.5, 0.25, and 0.1 V. Temperature-dependent variations of this circuitry are corrected by the software.

Only two of the eight available channels are used. One channel samples the output of the analog PCB and the other monitors an onboard temperature sensor, measuring the instrument's internal temperature. This

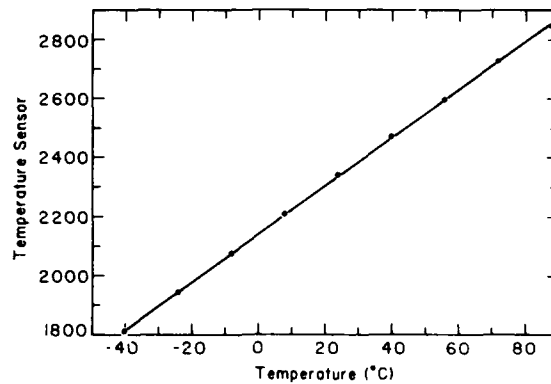


Figure 5. Output units of onboard temperature sensor.

measurement is required to allow the software to compensate for temperature-dependent variations in the analog PCB and the A/D board. The output from the internal sensor is proportional to the actual temperature. The relationship between the internal sensor output and temperature (°C) is shown in Figure 5.

Software

The software has two tasks. The first is to control the instrument itself, initializing the instrument when it is turned on, accessing the A/D converter, accepting information from the keyboard, displaying information, etc. The second is to accurately determine the temperature of the thermistor. It does this by applying the compensations required because of the instrument's temperature, calculating the thermistor's true resistance, and then solving the Steinhart-Hart equation. Appendix A contains listings and flowcharts for the software.

The software module that accomplishes the first task is labeled METER (Fig. A1a). METER was written in the higher level language PLM80. This language is very convenient for interfacing the several boards that form the instrument and it is especially useful for testing and manipulating the bits of data that must be transferred between the boards.

The software module that accomplishes the second task is labeled TMPCLC (Fig. A1b). This module is called as a subroutine by METER. TMPCLC was written in the higher level language PASCAL because it would have been impossible to do the required calculations with PLM80's 16-bit integer arithmetic. PASCAL, on the other hand, is very good for mathematical calculations but is difficult to use for interfacing boards.

As mentioned earlier, the instrument was developed on a National Semiconductor Corporation Starplex II microprocessor development system. However, all of the Starplex system calls, file structure subroutines and I/O subroutines were eliminated from the run-time library. This saves a great deal of memory space.

Combining the PASCAL and PLM80 languages presented special problems, including data transfer, duplicate function names and other interfacing problems. To facilitate data transfer, all information is passed between METER and TMPCLC through dedicated memory locations. This eliminates any need for matching variable types between the languages. Debugging is made easier when we know where each variable is stored.

METER

When the instrument is turned on, the program begins running METER. The first function is to initialize the I/O ports, the stack pointer and the PASCAL pointers. The A/D circuit board is also initialized. The program then repeatedly samples the A/D board. Sampling the A/D board for a valid reading is a two-step process. First, the programmable op-amp is set to the highest amplification, and second, a reading is taken on this range and checked for an over-range indication. If it is not over-range, the program returns to the main loop. If the reading was over-range the op-amp is set to the next lower amplification. The A/D board is sampled and checked again for an overrange condition. This process of reducing the amplification continues through each of the six ranges until a valid reading is obtained. This procedure is used to keep the number output from the A/D board as large as possible without going over-range, thus providing the maximum accuracy of the instrument. When a valid reading has been obtained, TMPCLC is called and the reading is converted to a temperature.

The keyboard is sampled within the range selection loop. If the 'F'-key is being pressed, the command interpreter subroutine labeled FUNCTION is called. All other keys will be ignored. There are two things the user may do while in FUNCTION. The thermistor calibration constants in use may be displayed or new ones entered (refer to Appendix B for more detailed operating instructions).

TMPCLC

The main function of TMPCLC is to convert the raw number output from the A/D converter to a temperature reading. First, TMPCLC reads the ther-

mistor constants from the EAROM and combines the digits to form floating point numbers. It uses a lookup table to determine the op-amp's voltage range and the range-dependent temperature compensation coefficients.

The first compensation is applied to the full-scale voltage of the A/D board at the range selected. Through experimentation, described in the next section, we found that the full-scale voltage could be described as

$$VFS_n = a_n + b_n T + c_n T^2 + d_n T^3 \quad (2)$$

where VFS_n is the full-scale voltage of the n th range, a_n , b_n , c_n , d_n are the temperature compensation coefficients associated with the n th range, and T is the instrument temperature, determined by the internal sensor.

Next, the raw binary output of the A/D board is corrected for temperature. This is done by the equation

$$Y = X_1 + m_n T + b_{on} \quad (3)$$

where X_1 is the raw output of the A/D board, m_n and b_{on} are constants, associated with the n th range, and T is the instrument temperature, determined by the onboard sensor. This equation in effect applies a temperature-dependent offset shift to the raw output of the A/D board. This offset was found through experimentation, again described in the next section.

The actual voltage measured by the A/D board is found as

$$MVADC = (Y/4095) * VFS \quad (4)$$

where $MVADC$ is the actual measured voltage, and 4095 is the total number of bits at full scale. Next, there is a known offset associated with the analog PCB, which is temperature-dependent. This offset, F_s , is determined as

$$F_s = a_{PCB} + b_{PCB} T + c_{PCB} T^2 + d_{PCB} T^3 \quad (5)$$

where a_{PCB} , b_{PCB} , c_{PCB} , d_{PCB} are the temperature compensation coefficients associated with the analog PCB, and T is the instrument temperature, determined by the onboard sensor. Finally, the true resistance of the thermistor, RES , can be calculated as

where RES is the thermistor resistance in ohms, 12.207E-06 is the constant current passed through the thermistor, and $MVTRUE = MVADC + F_s$.

Next, the Steinhart-Hart equation (eq 1) is used to convert the thermistor's resistance to a temperature. At present, the program will work only with thermistors with about 5 k Ω resistance at 0°C, as the exponents of the A, B, and C coefficients are declared in TMPCLC as constants.

INSTRUMENT ACCURACY

The following sources of possible error have been identified.

Errors external to the instrument

Probe lead resistance

The probe leads will add resistance in series with the thermistor. This will cause a decrease in the apparent temperature of the sample. The 6-ft 18-AWG (American Wire Gauge) stranded probe wire provided with the instrument has a resistance of roughly 30 m Ω . This results in an apparent temperature decrease of about 0.001°C at an ambient temperature of 20°C, which is the worst case.

Self-heating of the thermistor probe

Self-heating is the increase in temperature of the thermistor from the dissipation of electrical energy within the thermistor itself. Calculations based on dissipation constants for bead thermistors (Omega 1984) show that for the worst case of still air, the temperature error is only about +0.003°C. For a thermistor in a well-stirred oil bath, the error is only +0.0004°C in the worst case.

Calibration accuracy of the thermistors

Measurement error during the thermistor calibration is another factor but is beyond the scope of this report.

Uncertainty of the Steinhart-Hart equation

It has been shown that if the temperature span between any two adjacent calibration points is less than 50°C, the Steinhart-Hart equation will reproduce the actual temperature within 0.01°C (Yellow Springs Instruments, Inc. 1971).

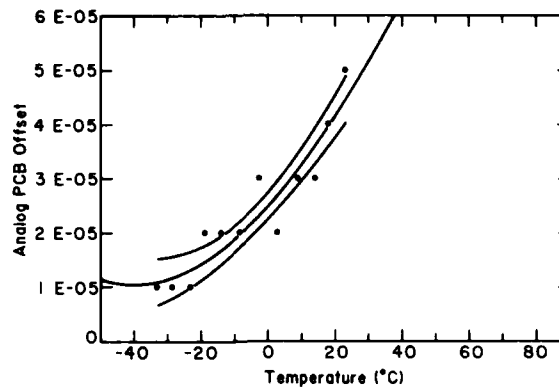


Figure 6. Analog PCB offset temperature compensation (center curve is eq 5, with 90% confidence band shown.)

Internal Instrument errors

The remaining sources of error are associated with the instrument itself. The individual errors have been characterized and, where possible, are corrected by the software. The temperature correction equations in TMPCLC play an important part in the instrument. Without them its accuracy would be far less. The following procedure was used to determine the temperature-dependent variation of the A/D converter board and the analog PCB.

Each was put into a cold chamber. The chamber's temperature was varied while the input to each board was held constant. The board output was measured and a regression done on the data relating the board's output to its temperature.

Temperature effects on analog PCB gain and offset

Through the tests described above we found that the offset (F_g) of the analog PCB was temperature dependent. Equation 5 determines the temperature-dependent offset of the resistance-to-voltage circuit. We found that offset error is much more significant than gain and nonlinearity errors. The latter two are small enough to be ignored (see Fig. 6 for the calibration curve).

Long-term stability of the analog PCB

The long-term stability of the circuit is both unknown and uncorrectable. The user should keep in mind, though, that most electronic instruments should be recalibrated at least once a year.

Accuracy of the PASCAL math package

National Semiconductor's PASCAL math package uses 24-bit floating point numbers internally. This maintains about 7 digits of precision. The resulting error is insignificant.

Round-off of displayed temperatures

The resolution of the liquid crystal display is 0.01°C . This means that the temperatures must be rounded to the nearest 0.01°C before being displayed. This contributes up to $\pm 0.005^{\circ}\text{C}$ error. This error can be reduced only by using a display with more digits.

A/D nonlinearity

The analog-to-digital converter's nonlinearity is uncorrectable and contributes $\pm 1/2$ bit to the instrument error.

A/D quantizing error

Quantizing error is present whenever there is converting between analog and digital. The uncertainty is always $\pm 1/2$ of the least-significant-bit of the converter.

A/D input offset current

The analog-to-digital converter's offset is affected by the output impedance of the previous stage. A higher output impedance results in more offset. In this case the offset is negligible because the op-amp in the previous stage has a very low output impedance.

A/D nonideal gain and offset

Equation 2 compensates for the temperature-dependent gain change of the A/D converter. The nominal full-scale voltage ranges are 5.0, 2.5, 1.0, 0.5, 0.25, and 0.1 V; there is a set of coefficients for each of the six voltage ranges. The program uses a table to select the proper set of coefficients. Figure 7 shows the raw data and regression plot for each voltage range.

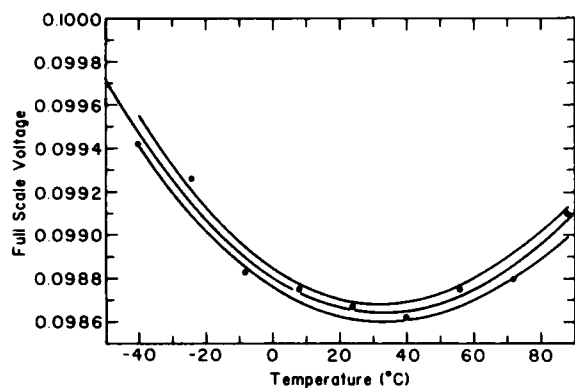
The raw output of the A/D conversion is corrected for offset shift by eq 3. This offset error is temperature dependent and can be approximated by a linear equation.

A/D long-term stability

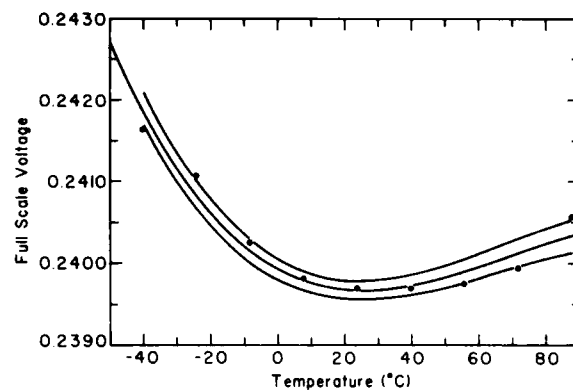
The long-term stability of the circuit is both unknown and uncorrectable. The circuit board has not been in use long enough to be evaluated over the long term. The user should keep in mind though that most electronic instruments should be recalibrated at least once a year.

Instrument error analysis

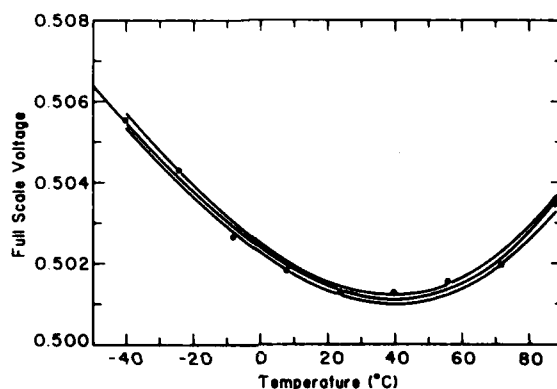
An error analysis was performed for two assumed field conditions. The first was to keep the thermistor's temperature at 0°C while varying the



a. 0.1-V range



b. 0.25-V range



c. 0.5-V range.

Figure 7. Calibration curves for A/D converter (center curve is eq 2 with 90% confidence band shown.)

Table 1. Thermistor at 0°C with instrument temperature varied.

Error source	Temperature (°C)											
	20	15	10	5	0	-5	-10	-15	-20	-25	-30	-35
Display round-off	± 0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64
A/D nonlinearity	± 0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
A/D quantizing	± 0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
A/D non-ideal gain	± 1.60	1.64	1.67	1.74	1.76	1.93	2.04	2.18	2.31	2.45	2.61	2.76
Analog PCB offset	± 0.18	0.14	0.12	0.10	0.10	0.10	0.10	0.11	0.12	0.14	0.17	0.19*
Total bits	± 3.42	3.42	3.43	3.48	3.50	3.67	3.78	3.93	4.07	4.23	4.42	4.59
Equivalent °C	± 0.027	0.027	0.027	0.027	0.027	0.029	0.030	0.031	0.032	0.033	0.035	0.036

*Estimated

Table 2. Thermistor and instrument at the same temperature.

Error Source	Temperature (°C)											
	20	15	10	5	0	-5	-10	-15	-20	-25	-30	-35
Display round-off	± 0.26	0.32	0.40	0.50	0.64	0.81	0.42	0.53	0.69	0.90	0.59	0.78
A/D nonlinearity	± 0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
A/D quantizing	± 0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
A/D non-ideal gain	± 1.60	1.64	1.67	1.74	1.76	1.93	2.37	2.53	2.69	2.85	1.40	1.48
Analog PCB offset	± 0.18	0.14	0.12	0.10	0.10	0.10	0.10	0.11	0.12	0.14	0.17	0.19*
Total bits	± 3.04	3.10	3.19	3.34	3.50	3.84	3.89	4.17	4.50	4.89	3.16	3.45
Equivalent °C	± 0.059	0.049	0.040	0.033	0.027	0.024	0.047	0.039	0.033	0.027	0.027	0.022

* Estimated

instrument's temperature. This condition corresponds to water temperature measurements made in the field. The instrument accuracy was calculated at 5°C intervals over the -35° to 20°C range (Table 1). The second condition was to have both thermistor and the instrument temperature the same. The instrument accuracy was again calculated at 5°C intervals over the same -35° to 20°C range (Table 2). This condition corresponds to air temperature measurements made in the field.

The main source of error was the uncertainty associated with regression equations describing the temperature-dependent variations of the A/D board's full-scale voltage. The uncertainty of these regressions, found using the 90% confidence bands, could be reduced by taking more calibration data for the A/D board within the operating temperature range of the instrument.

The error calculations in Table 3 are for CRREL thermistor serial no. 1805, which has a resistance of 5931.5 Ω and a change of 255 $\Omega/^\circ\text{C}$ at 0°C. Each error was converted to an equivalent number of bits at the A/D board then all the bits were summed to produce the total system error. The total system error in bits was changed to an equivalent temperature. The total system error describes the error band about the actual temperature.

Table 3 contains the results of a calibration done on 9 and 10 January 1985. The instrument was placed in a cold chamber and connected to a known resistance. The temperature of the chamber was set and the instrument allowed to equilibrate at that temperature. The known resistance was used to simulate thermistor no. 1805. The instrument temperatures were held constant at four temperatures between -18.3° and 24.0°C. For each

Table 3. Calibration on 9 and 10 January 1985 (°C).

Simulated	Error*	Simulated	Error*
Ambient temperature 24.0°C			
-1.00	0.00	1.00	0.00
-0.50	0.00	0.50	0.00
-0.40	0.00	0.40	-0.01 0.00
-0.30	0.00	0.30	-0.01 0.00
-0.20	0.00	0.20	-0.01 0.00
-0.10	0.00	0.10	0.00 -0.01
0.00	0.00		
Ambient temperature 3.9°C			
-1.00	-0.03	1.00	-0.01 0.00
-0.50	-0.01	0.50	-0.01 0.00
-0.40	-0.02	0.40	-0.01 -0.02
-0.30	0.00 -0.01	0.30	-0.01
-0.20	0.00 -0.01	0.20	-0.01 0.00
-0.10	-0.02 -0.01	0.10	-0.01
0.00	-0.01		
Ambient temperature -8.3°C			
-1.00	0.00 0.01	1.00	0.01 0.00
-0.50	0.00	0.50	0.00 0.01
-0.40	0.00	0.40	0.00
-0.30	0.00	0.30	0.00
-0.20	0.00	0.20	0.00
-0.10	0.00	0.10	0.00
0.00	0.00		
Ambient temperature -18.4°C			
-10.0	-0.02	10.0	0.04
-1.00	0.01	1.00	0.01
-0.50	0.01	0.50	0.01
-0.40	0.01 0.00	0.40	0.01
-0.30	0.01	0.30	0.00 0.01
-0.20	0.01 0.00	0.20	0.01 0.02
-0.10	0.01	0.10	0.01 0.02
0.00	0.01 0.00		

*Two listings means that the LCD was continually shifting between them.

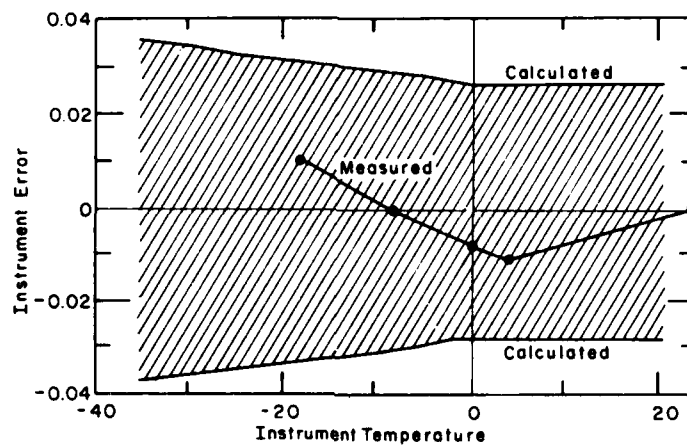


Figure 8. Error analysis (°C).

instrument temperature the simulated thermistor temperatures ranged from -1.0° to 1.0°C .

Figure 8 shows the comparison between the calculated and measured errors with the thermistor held at 0°C while the instrument temperature varied. As can be seen, the measured error was considerably less than the theoretical error.

SUMMARY

The following points would improve the instrument's performance.

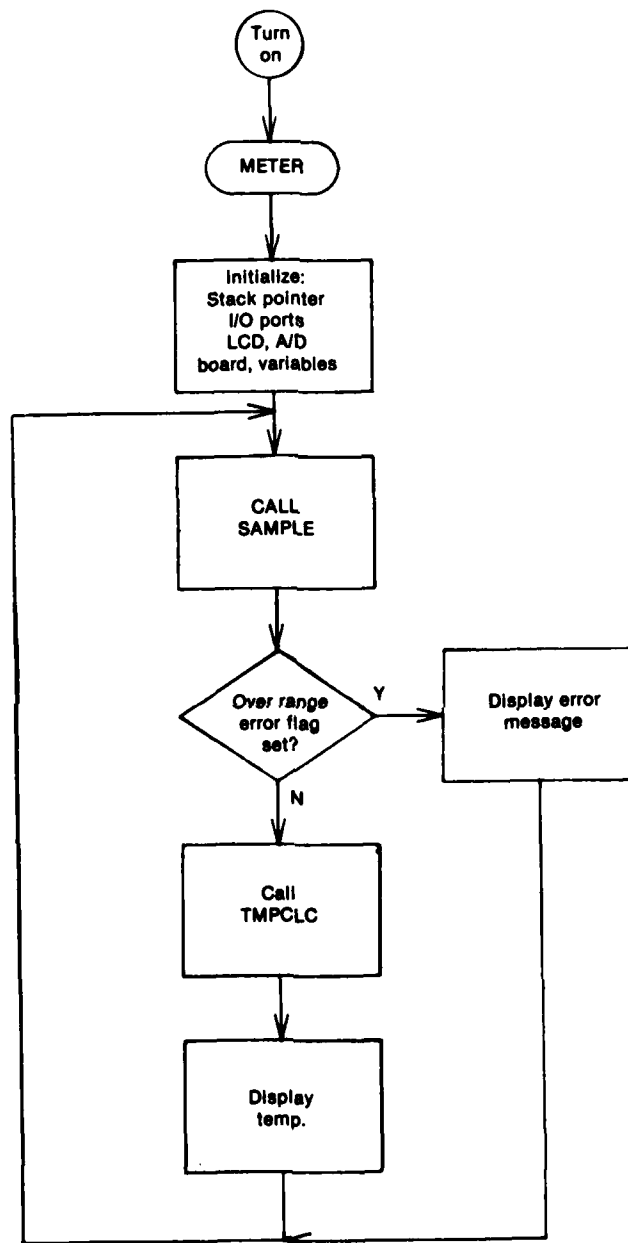
1. Use a self-timed EAROM. This would eliminate some modifications to the memory board and the processor board. The time delay circuit could also be eliminated.
2. Add a low battery voltage indication on the display. Two of the available A/D channels could be used to monitor the battery pack voltages directly. A flashing indication would then alert the user to a low battery condition.
3. Reduce the number of battery packs by using a DC/DC converter to develop the needed voltages.
4. Use a 14 or 16-bit A/D for greater accuracy.
5. Add an indicator that shows when the battery pack is indeed charging.
6. Set the gain of the instrumentation op-amp to 10 rather than 1. This can be done easily by grounding the proper instrumentation amplifier pin. This will allow the use of four ranges rather than two on the A/D, improving the accuracy on the upper end of the temperature range.

7. Take more calibration data (especially for the A/D PCB) within the ambient temperature range of interest. This will decrease the uncertainty associated with the regression equations.

LITERATURE CITED

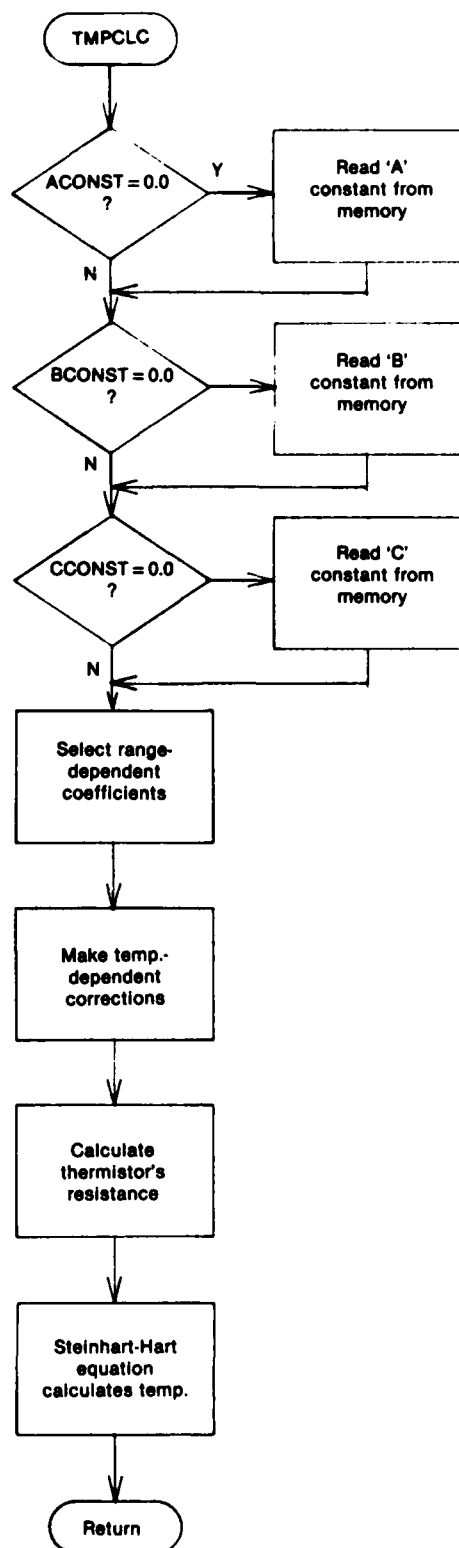
- Yellow Springs Instruments Inc. (1971) Thermistor Equation. Yellow Springs, Ohio: YSI, Industrial Division.
- Stout, D.F. and M. Kaufman (1976) Handbook of Operational Amplifier Circuit Design. New York: McGraw-Hill.
- Steinhart, J.S. and S.R. Hart (1968) Calibration curves for thermistors. Deep Sea Research, 15: 497.
- Omega (1984) Temperature Measurement Catalog. Stamford, Connecticut: Omega Engineering, Inc., p. f-9.

APPENDIX A: PROGRAMS



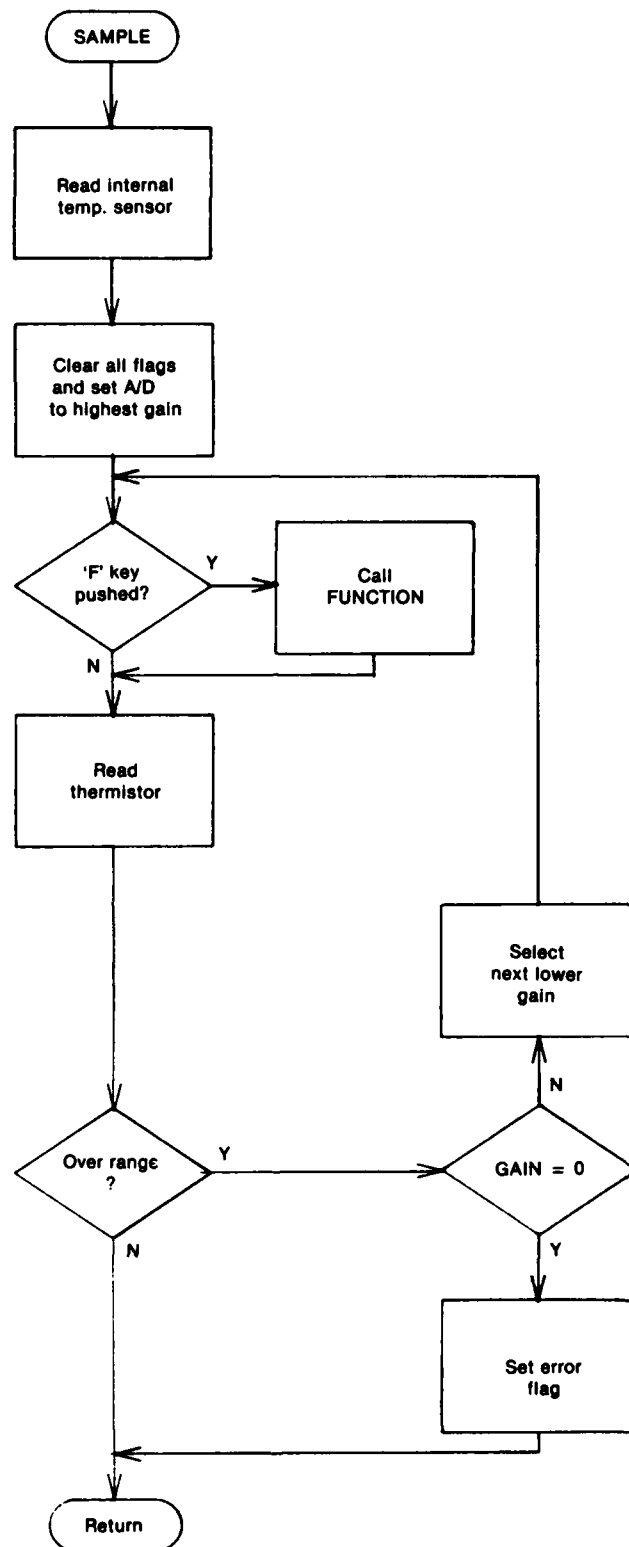
a. METER

Figure A1. Program flow charts.



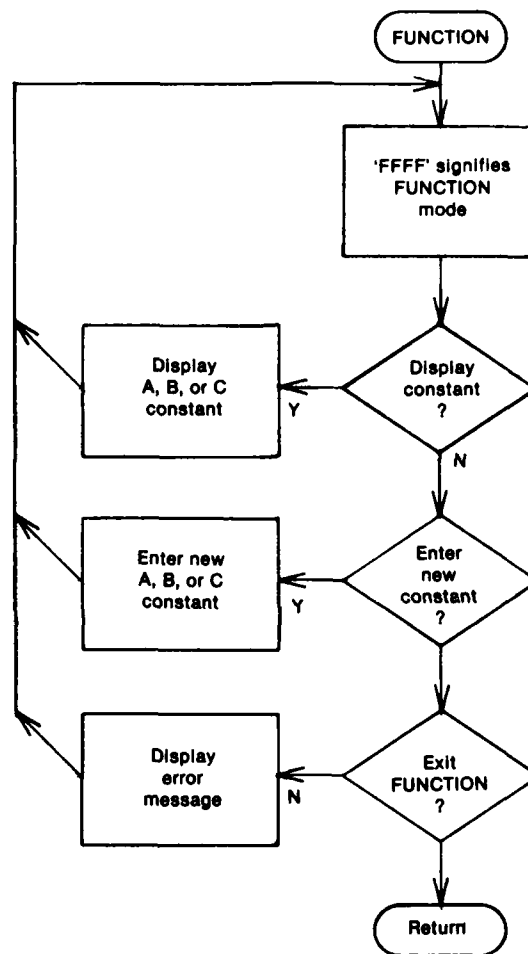
b. TMPCLC

Figure A1 (cont'd). Program flow charts.



c. SAMPLE

Figure A1 (cont'd).



d. FUNCTION.

Figure A1 (cont'd). Program flow charts.

/*

>>> PETER <<<

Written by Gary Trachier, USA CRREL
Last Update: 05 January 1985

This program handles all of the input and output of the UP system
and interfaces to the arithmetic program. The hardware requirements
are the following:

Onset Computer MEM-88 16K memory board
Onset Computer CPU-88C-1 CPU board
Quartic Systems QADC-12 12-bit A/D board
CRREL prototype Keyboard PCB
CRREL prototype LCD PCB

Some of the above circuit boards have been modified. Refer to the
project notebook for specific information.

CODE: 4000-7FFF
STACK: 1900-1FFF
HEAP: 190C-1FFF

IT APPEARS THAT PASCAL DOES NOT USE ANY SPACE
STARTING AT 1900. ONLY 50-100 BYTES ARE USED
BY THE STACK AT ANY ONE TIME. */

DATA: 120C-18FF

*/
PETER:DO;

```
DECLARE SIGN          BYTE;          /* RETURN SIGN OF READING FROM CHNLCONVRT */
DECLARE KEYCODE        BYTE;          /* ACTUAL KEY INPUT C->F */
DECLARE LOCATION       ADDRESS;
DECLARE CONSTANT       BASED LOCATION BYTE; /* DUMMY TEMPLATE FOR CONSTANTS */
DECLARE CVERTUNDE      BYTE;          /* OVER/UNDER RANGE INDICATION FROM 'SAMPLE' */
                                           /* 1CCC-0000=OVER RANGE */
                                           /* 0CCC-0001=UNDER RANGE */
                                           /* 0CCC-CCCC=OK (IN SOUNDS) */
```

```
DECLARE TEMPLO        BYTE AT (100CH);
DECLARE TEMPHI        BYTE AT (1001H);
DECLARE TEMPSGN        BYTE AT (1002H);
DECLARE ZROFLAG        BYTE AT (1005H);
DECLARE CDRA          BYTE AT (1094H);
DECLARE CORB          BYTE AT (1095H);
DECLARE CORC          BYTE AT (1096H);
DECLARE PORTA          BYTE AT (1097H);
DECLARE PORTB          BYTE AT (1098H);
DECLARE PORTC          BYTE AT (1099H);
DECLARE FORTC          BYTE AT (10B2H);
DECLARE AFLAG          BYTE AT (7FFFH);
DECLARE EFLAG          BYTE AT (7FFFH);

/* 1=ZERO, THE OFFSET TO +0.01C, 0=NORMAL OPERATION */
/* DCR, PORT A (LCD DATA IN LOWER NIBBLE) */
/* DCR, PORT B (LCD OP, SIGN, /CS, DIGIT SELECT) */
/* DCR, PORT C (KEYSCARD) */
/* ADDRESS OF PORT A */
/* ADDRESS OF PORT B */
/* ADDRESS OF PORT C */
/* EEPROM FLAG SHOWING THAT A-CONSTANT IS INITIALIZED */
/* EEPROM FLAG SHOWING THAT B-CONSTANT IS INITIALIZED */
```

```
DECLARE CFLAG          BYTE AT (7FFFH); /* EEPROM FLAG SHOWING THAT C-CONSTANT IS INITIALIZED */
DECLARE DISPLAY        ADDRESS;          /* USED TO SEND DATA TO LCD */
DECLARE ADDRNG         ADDRESS;          /* RAW RESULT OF A/D CONVERSION INCL. FLAGS */
DECLARE INDEX          ADDRESS;          /* G.P. LOOP COUNTER */
DECLARE CUMMY          ADDRESS;          /* G.P. VARIABLE */

DECLARE CFFSETBASE     LITERALLY 7824H; /* BASE OF THE ZERO OFFSET IN EEPROM */
DECLARE ABASE          LITERALLY 782CH; /* BASE OF A-CONSTANT */
DECLARE BBASE          LITERALLY 782EH; /* BASE OF B-CONSTANT */
DECLARE CBASE          LITERALLY 782FH; /* BASE OF C-CONSTANT */
DECLARE SETUPA         LITERALLY 1111-1111H; /* DIRECTION SETUP, PCRT A (LCD DATA) */
DECLARE SETUPB         LITERALLY 1111-1111H; /* DIRECTION SETUP, PCRT B (LCD CONTRL & SIGNS) */
DECLARE SETUPC         LITERALLY 00CC-0000H; /* DIRECTION SETUP, PCRT C (KEYBOARD) */
DECLARE ZRO            LITERALLY 0CH;
DECLARE TACKTCTP       LITERALLY 1FFFH; /* TCP OF STACK MEMORY */
DECLARE FORFEVER       LITERALLY WHILE 1;
DECLARE KBD            LITERALLY PORTC;
DECLARE KBDMSK         LITERALLY 00FF-1111H; /* LOCATION OF KEYBOARD INTERFACE */
DECLARE KBDVAL         LITERALLY 00FF-0000H; /* DATA MASK FOR KEYBOARD INPUTS */
DECLARE ADDATALO       LITERALLY 2CH; /* DATA VALID BIT MASK FOR KEYBOARD INTERFACE */
DECLARE ADDATAHI       LITERALLY 2CH; /* READ PORT FOR A/D LOW ORDER BYTE */
DECLARE AUSTATUS       LITERALLY 2CH; /* READ PORT FOR A/D HIGH ORDER BYTE */
DECLARE ADPMF          LITERALLY 2CH; /* READ PORT FOR A/D STATUS */
DECLARE ADGAINCHAL     LITERALLY 2CH; /* WRITE PORT FOR A/D POWER & RESET CONTROL */
DECLARE FOWERON        LITERALLY 00CC-0001H; /* WRITE PORT FOR A/D CHANNEL & GAIN CONTROL */
DECLARE FOWEROFF       LITERALLY 00CC-0000H; /* TURN QADC-12 POWER ON */
DECLARE RESETON        LITERALLY 00CC-0001H; /* TURN QADC-12 POWER OFF */
DECLARE RESETOFF       LITERALLY 00CC-0000H; /* ACTIVATE RESET ON QADC-12 */
DECLARE RESETOFF       LITERALLY 00CC-0010H; /* CLEAR RESET ON QADC-12 */
```

CCLOSTART: PROCEDURE INTERRUPT G; /* ROUTINE FOR 'RESET' & RSTO */
GOTO MAIN; /* GOTO START OF MAIN MODULE */
END CCLOSTART;

TEMPCLC: PROCEDURE EXTERNAL; /* PASCAL RESISTANCE TO TEMPERATURE ROUTINE */
END TEMPCLC;

BEGXCC: PROCEDURE EXTERNAL; /* INIT. THE PASCAL INTERNAL POINTERS */
END BEGXCC;

CAA: PROCEDURE (NUM) BYTE; /* CONVERT 8-BIT BINARY TO 2-DIGIT BCD */
DECLARE NUM BYTE;
DECLARE LSD BYTE;
DECLARE PSD BYTE;

LSD=NUM MOD 100;
PSD=(NUM-LSD)/100;
RETURN SPL(MSC,4) OR LSD;
END CAA;

ZRCCONST: PROCEDURE (CHCODE); /* ZEROS ONE OF THE PASCAL-CALCULATED CONSTANTS IN RAM */
DECLARE CHCODE BYTE; /* INPUT SHOULD HAVE 'A', 'B', OR 'C' IN THE LOWER NIBBLE */
DECLARE INDEX BYTE; /* G.P. LOOP COUNTER */
DECLARE LOCATION ADDRESS; /* ADDRESS TO BE ZEROED */
DECLARE CONST BASED LOCATION BYTE; /* ONE-BYTE TEMPLATE FOR CLEARING MEMORY */

```

DECLARE MASK LITERALLY '0000_1111B'; /* LC-NIBBLE MASK */
DECLARE ACONST LITERALLY '1000H'; /* BASE FOR 4-BYTE CALCULATED A-CONSTANT */
DECLARE BCONST LITERALLY '1000H'; /* BASE FOR 4-BYTE CALCULATED B-CONSTANT */
DECLARE CCONST LITERALLY '1000H'; /* BASE FOR 4-BYTE CALCULATED C-CONSTANT */

IF (CMCCCE AND MASK)=0AH THEN /* CLEAR THE A-CONSTANT */
  LOCATION=ACONST;
IF (CMCCCE AND MASK)=0BH THEN /* CLEAR THE B-CONSTANT */
  LOCATION=BCONST;
IF (CMCCCE AND MASK)=0CH THEN /* CLEAR THE C-CONSTANT */
  LOCATION=CCONST;
DO INDEX 1 TO 4; /* CLEAR THE 4-BYTE REAL NUMBER */
  CONST=ZRC; /* CLEAR THE BYTE */
  LOCATION=LOCATION+1; /* POINT TO THE NEXT BYTE */
END;
END ZROCCNST;

SETCLR: PROCEDURE (PORT, BITMSK, VALUE); /* PORT BIT SET/CLR ROUTINE */
  /* 1 FOR PORT A, 2 FOR PORT B, 3 FOR PORT C */
  /* BITS TO BE SET OR CLEARED ARE 1, ALL OTHERS ARE 0 */
  /* BITS CORRESPONDING TO THOSE SET IN BITS SHOULD BE THE DESIRED DATA, IE 1 OR 0, ALL OTHERS ARE IRRELEVANT */
  DECLARE PORT BYTE;
  DECLARE BITMSK BYTE;
  DECLARE VALUE BYTE;

  DECLARE SETDATA BYTE;
  DECLARE CLRDATA BYTE;
  PORTASET=PORT AT (1C8CH); /* PORT A BIT SET ADDRESS */
  PORTACLR=PORT AT (1C8SH); /* PORT A BIT CLEAR ADDRESS */
  PORTBSET=PORT AT (1C90H); /* PORT B BIT SET ADDRESS */
  PORTBCLR=PORT AT (1C90H); /* PORT B BIT CLEAR ADDRESS */
  PORTCSET=PORT AT (1C9EH); /* PORT C BIT SET ADDRESS */
  PORTCCLR=PORT AT (1C9EH); /* PORT C BIT CLEAR ADDRESS */

  SETDATA=(BITMSK AND VALUE); /* BITS TO BE SET ARE 1 */
  CLRDATA=(BITMSK XOR VALUE AND BITMSK); /* BITS TO BE CLEARED ARE NOW 1 */
  DO CASE (PORT-1); /* PORT A */
    PORTASET=SETDATA;
    PORTACLR=CLRDATA;
  END;
  DO; /* PORT B */
    PORTBSET=SETDATA;
    PORTBCLR=CLRDATA;
  END;
  DO; /* PORT C */
    PORTCSET=SETDATA;
    PORTCCLR=CLRDATA;
  END;
END SETCLR;

LCDBYTE: PROCEDURE (DIGIT, CNEDIGIT, DECIMAL, HBAR, VBAR, VALUE); /* OUTPUTS ONE DIGIT OF DATA & VARIOUS
  /* CP, BARS ETC */
  /* DIGIT SELECT (0-3) */
  /* ONEDIGIT ANNUNCIATOR, UNUSED */
  /* DECIMAL POINT, 1=ON, 0=OFF */
  /* HORIZONTAL BAR */

  DECLARE DIGIT BYTE;
  DECLARE CNEDIGIT BYTE;
  DECLARE DECIMAL BYTE;
  DECLARE HBAR BYTE;

  DECLARE VBAR BYTE;
  DECLARE VALUE BYTE;
  DECLARE DUMMY BYTE;

  DECLARE CNEMSK LITERALLY '0000_0001B'; /* HALF-DIGIT '1' MASK */
  DECLARE CPMSK LITERALLY '0000_0001B'; /* DECIMAL POINT MASK */
  DECLARE HPMASK LITERALLY '0000_0001B'; /* HORIZONTAL BAR MASK */
  DECLARE VPMASK LITERALLY '0000_0001B'; /* VERTICAL BAR MASK */
  DECLARE VALMSK LITERALLY '0000_1111B'; /* NUMERIC DATA MASK */

  VALUE=VALUE AND VALMSK; /* SAVE ONLY THE LOWER NIBBLE */
  DIGIT=(DIGIT XOR 0000_0011B); /* CHANGE TO 7211'S NUMBERING SYSTEM */
  PORTA=VALUE; /* SEND THE HEXADECIMAL NUMBER TO PORTA */
  DUMMY=SHL(DIGIT, 1); /* SET UP THE DIGIT SELECT BITS */
  CALL SETCLR(2, 0000_0001B, 0000_0000B); /* THEN SEND AN ACTIVE-LCM PULSE */
  CALL SETCLR(2, 0000_0001B, 0000_0001B); /* TC DS ON THE 7211 */
  DUMMY=SHL(CNEDIGIT AND CNEMSK, 7); /* UPDATE '1' DIGIT */
  CALL SETCLR(2, 1000_0000B, DUMMY); /* UPDATE DECIMAL POINT */
  DUMMY=SHL(HBAR AND HPMASK, 5); /* UPDATE HORIZONTAL BAR */
  CALL SETCLR(2, 0001_0000B, DUMMY); /* UPDATE VERTICAL BAR */
  DUMMY=SHL(VBAR AND VPMASK, 4);
  CALL SETCLR(2, 0001_0000B, DUMMY);

END LCDBYTE;

MSDELAY: PROCEDURE (MS); /* 1C-MS DELAY ROUTINE */
  DECLARE COUNT ADDRESS;
  DECLARE COUNT2 ADDRESS;
  DECLARE MS BYTE;

  DECLARE MSBLY LITERALLY '005CH'; /* LCCP CONSTANT */
  DO COUNT=1 TO MS;
    DO COUNT2=0 TO MSBLY;
      FN; /* 1C-MS BLOCK */
    END;
  END;
END MSDELAY;

DELAY: PROCEDURE (SECONDS); /* 1 SECOND DELAY ROUTINE */
  DECLARE SECONDS BYTE;
  DECLARE COUNT ADDRESS;
  DECLARE COUNT2 ADDRESS;

  DECLARE CLV LITERALLY '2550H'; /* LCCP CONSTANT */
  DO COUNT=1 TO SECONDS;
    DO COUNT2=0 TO CLV;
      FN; /* 1 SECOND BLOCK */
    END;
  END;

```

```

END;
END DELAY;

CHNLNVRT: PROCEDURE (CHANNEL, GAIN, MODE); /* DCES CONVERSION FOR A SINGLE CHANNEL & GAIN */
    DECLARE GAIN BYTE; /* 0-5 */
    DECLARE CHANNEL BYTE; /* 0-15 SINGLE-ENDED, 0-7 DIFFERENTIAL */
    DECLARE MODE BYTE; /* 0=DIFFERENTIAL, 1=SINGLE-ENDED */
    DECLARE CUMMY ADDRESS;

    DECLARE ACTIVE LITERALLY '0000-00013'; /* A/D STATUS, 1=CONVERTING 0=INACTIVE */
    DECLARE SIGNMSK LITERALLY '1000-00003'; /* 1=POSITIVE, 0=NEGATIVE */
    DECLARE DATAMSK2 LITERALLY '0001111-11111108'; /* DATA MASK WITH THE LOWER BITS AS "DON'T CARE" */

    IF MODE=0 THEN
        MODE=C00-C1-CC0B; /* DIFFERENTIAL */
    ELSE
        MODE=000-10-CC0B; /* SINGLE-ENDED */
    END IF;

    CNVRT2: OUTPUT(ACGAINCHNL)=SHL(GAIN,50) OR MODE OR CHANNEL; /* SETUP GAIN, MODE & CHANNEL THEN DO A CONVERSION */

    DO WHILE (INPUT(ADSTATUS) AND ACTIVE)=ZRC; /* WAIT FOR A/D STATUS BIT TO GO HIGH... */
    END;
    DO WHILE (INPUT(ADSTATUS) AND ACTIVE)=01H; /* ...THEN LG... */
    END;

    CUMMY=INPUT(ADCATAMI); /* GET THE UPPER BITS & FLAGS */
    SIGN=SHL(CUMMY AND SIGNMSK,1); /* EXTRACT THE SIGN BIT */
    CUMMY=SHL(CUMMY,8); /* SHIFT TO UPPER BYTE */
    CUMMY=CUMMY OR INPUT(ADCATALC); /* THEN GET THE LOWER BYTE */
    IF (ACDRNG AND DATAMSK2) <> (CUMMY AND DATAMSK2) THEN
        CO;
        ADRNG=CUMMY; /* SAVE THE LAST READING */
        GOTO CNVRT2; /* REPEAT UNTIL THE READINGS CLOSELY AGREE */
    END;
    ADRNG=CUMMY; /* RETURN THE STABILIZED VALUE */
END CHNLNVRT;

KEYIN: PROCEDURE; /* DEBOUNCED SINGLE KEY INPUT ROUTINE */
    DECLARE TMPKEY BYTE; /* TEMPORARY RAW KEYCODE STORAGE */
    DECLARE KBDSTBL(160) BYTE DATA /* CONVERSION TABLE FROM RAW TO REAL KEYCODE */
        (00H,01H,02H,03H,04H,05H,06H,07H,08H,
         09H,0AH,0BH,0CH,0DH,0EH,0FH);

    DECLARE MASK LITERALLY '0000-1111B'; /* MASK OUT UNUSED BITS */

    DO WHILE (KBD AND KBDVAL)=KBDVAL; /* WAIT FOR LAST KEY TO BE RELEASED */
    END;
    CALL MSSDELAY (10); /* DELAY FOR 10-MS */
    KEYIN1: DO WHILE (KBD AND KBDVAL)=ZRC; /* WAIT FOR NEXT KEY INPLT */
    END;
    TMPKEY=KBD; /* GRAB THE RAW KEYCODE FROM MEMORY-MAPPED KEYBOARD PCRT */
    CALL MSSDELAY (10); /* DELAY FOR 10-MS */

    IF TMPKEY<>KBD THEN
        GOTO KEYIN1; /* IF KEY BOUNCED THEN READ AGAIN */
    END KEYIN; /* LOOKUP THE ACTUAL KEYCODE AND RETURN TO CALLER */

LCD: PROCEDURE (ONEDIGIT, DECIMAL, HBAR, VBAR, VALUE);
    DECLARE CINDEX BYTE; /* COUNTER FOR DIGIT */
    DECLARE INTRX BYTE; /* LOOP COUNTER */
    DECLARE INTR2 BYTE; /* LOOP COUNTER */
    DECLARE CMDCODE BYTE; /* WILL BECOME DA, DB, DC, EA, EB, EC, OR EE */
    DECLARE BASE BYTE; /* POINTER FOR THE "CONSTANT" ARRAY */
    DECLARE BASE2 ADDRESS; /* POINTS TO BASE OF THE CONSTANT TO BE CHANGED/DISPLAYED */

    CALL LCDBYTE (INDEX, ONEDIGIT, DECIMAL, HBAR, VBAR, VALUE);
    VALUE=SHR(VALUE,4);
END LCD;

RTERRR: PROCEDURE INTERRUPT 1; /* THIS SHOULD START AT 0008H */
    CALL LCD (ZRC,ZRC,ZRC,ZRC,DEFEH);
    CALL DELAY (10);
    CALL LCD (ZRC,ZRC,ZRC,ZRC,000CH);
    CALL DELAY (10);
END RTERRR;

FUNCTION: PROCEDURE; /* INVOKED BY "F" KEY FROM KEYBOARD */
    DECLARE CONSTARRAY ARRAY(160) BYTE; /* ARRAY FOR ENTERING THE CONSTANTS */
    DECLARE INTRX2 BYTE; /* LOOP COUNTER */
    DECLARE INTR3 BYTE; /* LOOP COUNTER */
    DECLARE CMDCODE2 BYTE; /* WILL BECOME DA, DB, DC, EA, EB, EC, OR EE */
    DECLARE BASE22 BYTE; /* POINTER FOR THE "CONSTANT" ARRAY */
    DECLARE BASE222 ADDRESS; /* POINTS TO BASE OF THE CONSTANT TO BE CHANGED/DISPLAYED */

    DO FOREVER; /* DO UNTIL BROKEN OUT THROUGH "EE" COMMAND */
        KEYCODE=ZRC;
        CMDCODE=ZRC;
        DISPLAY=0FFFFH; /* PUT "FFFF" ON DISPLAY... */
        CALL LCD (ZRC,ZRC,ZRC,ZRC,DISPLAY); /* TO SIGNIFY "FUNCTION" MODE */
        CALL KEYIN; /* GET THE FIRST KEY */
        CMDCODE=SHL(KEYCODE,4); /* AND MOVE IT TO THE UPPER NIBBLE */
        DISPLAY=KEYCODE;
        CALL LCD (ZRC,ZRC,ZRC,ZRC,DISPLAY); /* KEY 1 TO LCD */
        CALL KEYIN; /* GET THE SECOND KEY */
        CMDCODE=CMDCODE OR KEYCODE; /* COMBINE UPPER AND LOWER NIBBLES */
        DISPLAY=CMDCODE;
        CALL LCD (ZRC,ZRC,ZRC,ZRC,DISPLAY); /* KEY 2 TO LCD */
        CALL MSSDELAY (500); /* 500-MS DELAY */

        DO CASE CMDCODE; /* NOW EXECUTE THE COMMAND */
            21H: /* DA */
            22H: /* DB */
            23H: /* DC */
            24H: /* DA */
            25H: /* DB */
            26H: /* DC */
            27H: /* DA */
            28H: /* DB */
            29H: /* DC */
            2AH: /* DA */
            2BH: /* DB */
            2CH: /* DC */
            2DH: /* DA */
            2EH: /* DB */
            2FH: /* DC */
            30H: /* DA */
            31H: /* DB */
            32H: /* DC */
            33H: /* DA */
            34H: /* DB */
            35H: /* DC */
            36H: /* DA */
            37H: /* DB */
            38H: /* DC */
            39H: /* DA */
            3AH: /* DB */
            3BH: /* DC */
            3CH: /* DA */
            3DH: /* DB */
            3EH: /* DC */
            3FH: /* DA */
            40H: /* DB */
            41H: /* DC */
            42H: /* DA */
            43H: /* DB */
            44H: /* DC */
            45H: /* DA */
            46H: /* DB */
            47H: /* DC */
            48H: /* DA */
            49H: /* DB */
            4AH: /* DC */
            4BH: /* DA */
            4CH: /* DB */
            4DH: /* DC */
            4EH: /* DA */
            4FH: /* DB */
            50H: /* DC */
            51H: /* DA */
            52H: /* DB */
            53H: /* DC */
            54H: /* DA */
            55H: /* DB */
            56H: /* DC */
            57H: /* DA */
            58H: /* DB */
            59H: /* DC */
            5AH: /* DA */
            5BH: /* DB */
            5CH: /* DC */
            5DH: /* DA */
            5EH: /* DB */
            5FH: /* DC */
            60H: /* DA */
            61H: /* DB */
            62H: /* DC */
            63H: /* DA */
            64H: /* DB */
            65H: /* DC */
            66H: /* DA */
            67H: /* DB */
            68H: /* DC */
            69H: /* DA */
            6AH: /* DB */
            6BH: /* DC */
            6CH: /* DA */
            6DH: /* DB */
            6EH: /* DC */
            6FH: /* DA */
            70H: /* DB */
            71H: /* DC */
            72H: /* DA */
            73H: /* DB */
            74H: /* DC */
            75H: /* DA */
            76H: /* DB */
            77H: /* DC */
            78H: /* DA */
            79H: /* DB */
            7AH: /* DC */
            7BH: /* DA */
            7CH: /* DB */
            7DH: /* DC */
            7EH: /* DA */
            7FH: /* DB */
            80H: /* DC */
            81H: /* DA */
            82H: /* DB */
            83H: /* DC */
            84H: /* DA */
            85H: /* DB */
            86H: /* DC */
            87H: /* DA */
            88H: /* DB */
            89H: /* DC */
            8AH: /* DA */
            8BH: /* DB */
            8CH: /* DC */
            8DH: /* DA */
            8EH: /* DB */
            8FH: /* DC */
            90H: /* DA */
            91H: /* DB */
            92H: /* DC */
            93H: /* DA */
            94H: /* DB */
            95H: /* DC */
            96H: /* DA */
            97H: /* DB */
            98H: /* DC */
            99H: /* DA */
            9AH: /* DB */
            9BH: /* DC */
            9CH: /* DA */
            9DH: /* DB */
            9EH: /* DC */
            9FH: /* DA */
            A0H: /* DB */
            A1H: /* DC */
            A2H: /* DA */
            A3H: /* DB */
            A4H: /* DC */
            A5H: /* DA */
            A6H: /* DB */
            A7H: /* DC */
            A8H: /* DA */
            A9H: /* DB */
            AAH: /* DC */
            ABH: /* DA */
            ACH: /* DB */
            ADH: /* DC */
            AEH: /* DA */
            AFH: /* DB */
            B0H: /* DC */
            B1H: /* DA */
            B2H: /* DB */
            B3H: /* DC */
            B4H: /* DA */
            B5H: /* DB */
            B6H: /* DC */
            B7H: /* DA */
            B8H: /* DB */
            B9H: /* DC */
            BAH: /* DA */
            BBH: /* DB */
            BCH: /* DC */
            BDH: /* DA */
            BEH: /* DB */
            BFH: /* DC */
            CH: /* DA */
            DH: /* DB */
            EH: /* DC */
            FH: /* DA */
        END CASE;
    END FOREVER;
END FUNCTION;

```

```

CO;
IF KEYCODE=0AH THEN
  BASE=ABASE; /* POINT TO THE A CONSTANT */
IF KEYCODE=0BH THEN
  BASE=BBASE; /* POINT TO THE B CONSTANT */
IF KEYCODE=0CH THEN
  BASE=CBASE; /* POINT TO THE C CONSTANT */
DO INCEX=0 TO 3; /* GROUP COUNTER */
  DC INCEX2=0 TO 3; /* DIGIT COUNTER */
  LCCATION=BASE+(4*INDEX)+(3-INDEX2); /* CALC. THE OFFSET */
  CALL LCDBYTE (INCEX2,ZRO,ZRO,ZRO,ZRO,CONSTANT); /* DISPLAY
    A GROUP OF 4
  END;
  CALL KEYIN; /* AND WAIT FOR LSFR TO PROMPT FOR THE NEXT GROUP OF 4
END;
END;
236: /* A */
237: /* B */
238: /* C */
CO;
IF KEYCODE=0AH THEN
  CO;
  AFLAG=OFFH; /* CLEAR THE BYTE */
  AFLAG=0AAH; /* THEN INIT. IT */
  BASE=ABASE; /* ENTER THE A CONSTANT */
END;
IF KEYCODE=0BH THEN
  DO;
  BFLAG=OFFH;
  BFLAG=0BBH;
  BASE=BBASE; /* ENTER THE B CONSTANT */
END;
IF KEYCODE=0CH THEN
  CO;
  CFLAG=OFFH;
  CFLAG=0CCH;
  BASE=CBASE; /* ENTER THE C CONSTANT */
END;
CO INCEX=0 TO 3;
CONST$ARRAY(INCEX)=KEYCODE; /* FILL FIRST 4 PLACES WITH A,B,C, OR D
END;
CO INCEX=4 TO 15;
CONST$ARRAY(INCEX)=ZRC; /* THEN THE REST WITH ZEROS. */
END;
CO INCEX=0 TO 3;
DISPLAY=SHL(DISPLAY,4) OR CONST$ARRAY(INCEX);
END;
BASE2=0; /* INIT. THE ARRAY PCINTER */
CO WHILE BASE2 < 110; /* DIGIT COUNTER */
  KEYCODE=CFH;
  CALL LCD (ZRC,ZRC,ZRC,ZRC,DISPLAY);
  DO WHILE (KEYCODE>9) AND (KEYCODE<>0BH); /* WAIT FOR 0-9 OR 'B'
    KEY INPUT */
  CALL KEYIN;
END;
IF KEYCODE<100 THEN /* DIGITS 0-9 */
  CO;
  CONST$ARRAY(BASE2+4)=KEYCODE; /* SAVE THE DIGIT */
  BASE2=BASE2+1; /* INC. THE ARRAY POINTER */
  DISPLAY=SHL(DISPLAY,40);
  DISPLAY=DISPLAY OR KEYCODE; /* PUT THE DIGIT ON THE
    DISPLAY */
END;
IF KEYCODE=0BH AND (BASE2>0) THEN /* BACKUP A DIGIT-AT-A-TIME */
  CO;
  BASE2=BASE2-1; /* DEC. THE ARRAY POINTER */
  CO INCEX=0 TO 3; /* BACKUP ONE DIGIT &
    RESTRUCTURE 'DISPLAY'
  DISPLAY=SHL(DISPLAY,4) OR
    CONST$ARRAY(BASE2+INDEX);
END;
END;
END;
CALL LCD (ZRO,ZRC,ZRC,ZRC,DISPLAY); /* SHOW THE LAST DIGIT */
CALL DELAY (10); /* 1-SECOND DELAY */
CO INCEX=0 TO 11; /* WRITE THE ARRAY INTO EEROM */
  LOCATION=BASE+INDEX; /* CALC. THE DESTINATION ADDRESS */
  CONSTANT=CFH; /* CLEAR THE EERCP BYTE */
  CONSTANT=CONST$ARRAY(INCEX+4); /* THEN WRITE THE ACTUAL DIGIT */
END;
CALL ZROCONST (CMGCODE); /* CLEAR THE REAL-TYPE TC ALERT PASCAL
  ROUTINE */
END;
238: RETURN; /* EE */ /* RETURN TO TEMPERATURE MEASURE MODE */
240: CO; /* FD */
  ZROFLAG=1; /* DC AN OFFSET ZERO (<0.C1C) NEXT TIME TYPCLC IS CALLED */
  CO LCCATION=OFFSETBASE TO OFFSETBASE+3;
  CONSTANT=CFH; /* ERASE THE EERCM SO A NEW OFFSET CAN BE WRITTEN */
END;
RETURN; /* GC BACK TO TEMPERATURE MEASURE MODE */
END;
OTHERWISE DO; /* THIS CODE IS FOR ALL INVALID COMMANDS */
  CALL LCD (ZRO,ZRC,ZRC,ZRC,DEEEE); /* SHOW EEEE AS AN ERROR CONDITION */
  CALL DELAY (10); /* WAIT FOR 1 SECOND */
  CALL LCD (ZRO,ZRC,ZRC,ZRC,000CH); /* THEN ZERO THE DISPLAY */
END;
END;
END;
END FUNCTION;
SAMPLE:PROCEDURE; /* DCES CONVERSION OF THERMISTOR & PCB TEMP. SENSOR */
  DECLARE GAIN ADDRESS AT (1C14H); /* 0->5 GAIN RANGE */
  DECLARE RONG ADDRESS AT (1CC3H); /* 16-BIT SPACE FOR A/D VALUE */
  DECLARE GOOD$PPL BYTE;

```

```

DECLARE AMRTMP ADDRESS AT (1C12H); /* SEND PCB TEMP. TO TMPCLC ROUTINE */
DECLARE CATAMSK LITERALLY '00001111 11111111'; /* KEEPS 12 DATA BITS, THROU OUT OV, SIGN ETC. */
DECLARE CVRFLW LITERALLY '0010000-00000000'; /* A/D OVERFLW BIT, 1=OVERFLW C=OK */
DECLARE NEGPCS LITERALLY '1000000-00000000'; /* A/D SIGN BIT, 1=PCS 0=NEG */
DECLARE CVRFLWFLAG LITERALLY '1000_0000'; /* OVER THE DYNAMIC RANGE OF THE OP-AMP & A/D SYSTEM */

CALL CHNLNVRT (0,2,1); /* READ THE PCB TEMP. SENSOR */
AMEYMP=ACRONG AND CATAMSK; /* MASK OUT THE FLAG BITS */
GAIN=5; /* START WITH MAXIMUM GAIN */
GOODSMPL=ZRO;
OVERSUNDR=ZRO;
CVRFLWFLAG=ZRO;
CC WHILE GOODSMPL=ZRO; /* LOOP 'TILL A GOOD READING OR THE LOWEST RANGE IS REACHED */

    KEYCOFF=ZRO;
    IF (KBD AND KBDVAL)=KBDVAL THEN /* GRAB THE KEY BEING PUSHED */
        KEYCOFF=KBD AND KBDMSK;
        IF KEYCODE=OFFH THEN /* GC INTO THE FUNCTION MODE */
            CALL FUNCTION;
        CALL CHNLNVRT (1,GAIN,0); /* READ THE THERMISTOR */
        PDNG=ACRONG AND CATAMSK; /* SAVE THE DATA BITS */
        IF (ACRONG AND OVRFLW)=ZRO THEN /* GCCD CONVERSION, NO OVERFLOW */
            ELSE
                CC;
                IF GAIN=ZRO THEN
                    DO;
                        OVERSUNDR=OVRFLWFLAG; /* FLAG AS AN OVER RANGE READING ON LOWEST GAIN RANG */
                        GOODSMPL=OFFH; /* SET FLAG SO WE CAN BREAK OUT OF LOOP */
                    END;
                ELSE
                    GAIN=GAIN-1; /* TRY THE NEXT LOWER GAIN SETTING */
                END;
            END;
        END;
    END;
END SAMPLE;

MAIN:DO; /* MAIN CODE SECTION */
    /* SET UP THE STACK PCINTER */
    CALL REG700; /* SET UP THE PASCAL INTERNALS */
    CORA=SETLPA; /* SET UP THE NSC870 I/O PORTS */
    CORB=SETLPA;
    CORC=SETLPC;
    PORTA=ZRO; /* CLEAR ANY GLITCHES STORED IN LATCH */
    PORTB=1111 0001; /* SET UP DIGIT SELECT & SPECIAL CONTROL BITS ON LCD */
    CUMHY=PORTC; /* CLEAR THE KEYBOARD LATCH */
    CALL LCD (1,1,1,1,8888H); /* TURN ALL DISPLAY SEGMENTS ON... */
    CALL DELAY (10); /* AND WAIT 1 SECOND. */
    OUTPUT(ACPMR)=POWERON; /* POWER UP THE CADC-12 */
    CALL M55DELAY(50); /* WAIT FOR 50MS... */
    OUTPUT(AEPMR)=POWERON OR RESETCF; /* 1, THEN CLEAR THE RESET BIT */
    CUMHY=INPUT(ADSTATUS); /* CLEAR THE CADC-12 INTERRUPT BIT */
    ZROFLAG=ZRO; /* CLEAR THE AUTC ZERC FLAG */

    CALL ZROCONST (0AH); /* CLEAR THE A CALCULATED CONSTANT */
    CALL ZROCONST (0BH); /* CLEAR THE B CALCULATED CONSTANT */
    CALL ZROCONST (0CH); /* CLEAR THE C CALCULATED CONSTANT */

    DO WHILE (AFLAG<>CAAH) OR (BFLAG<>0BBH) OR (CFLAG<>0CCH);
        CALL FUNCTION; /* FORCE ENTRY OF NEW CONSTANTS IN NEW EEROM */
        DO LOCATION=CFFSETBASE TO OFFSETBASE+3;
            CONSTANT=ZRO; /* ZERO THE CFFSET IF A NEW EEROM */
        END;
    END;

    DO FOREVER;
        CALL SAMPLE; /* GET A READING */
        IF OVERSUNDR>ZRO THEN /* DISPLAY AN ERROR INDICATION */
            CALL LCD (ZRO,ZRO,ZRO,ZRO,0EEEH);
        ELSE
            CC;
            CALL TMPCLC; /* CONVERT RESISTANCE TO TEMPERATURE */
            DISPLAY=DAA(TEMPHI); /* COMBINE THE UPPER... */
            DISPLAY=SHL(DISPLAY,2C) OR DAA(TEMPL0); /* AND LOWER 2 DIGITS */
            CALL LCD (ZRO,1,1,NOT TEMPSGN,DISPLAY); /* AND OUTPUT W/ OP & SGN TO DISPLAY */
        END;
    END;
END MAIN;
END METER;

```

(* >>> TMPCLC (V 1.0) <<<

Written by Gary Trachler, USA, CRREL
Last updated: 01 January 1985

This program interfaces to the x86 code named "METER". It is called as an untyped procedure in METER. This routine uses the 12-bit number from the A/D, the thermistor constants that are stored in EARGM, and the gain setting of the instrumentation op-amp as inputs. It then uses the Steinhart, Hart equation to calculate the temperature. The temperature is returned to the main code in TEMPLO and TEMPHI. TEMPLO is an 8-bit representation of the two digits to the right of the decimal point. This will range from 0 to 99. TEMPHI is a value from 0 to 99 corresponding to the two digits to the left of the decimal point. The sign of the temperature is returned in TEMPSGN.

This module must be compiled by the Stardex Pascal compiler. It then must be linked to the skeleton Pascal library and the relocatable code from "METER". Refer to the compilation and linkage instructions for specifics.

*)
PROGRAM CUMMY;

```
VAR ERTLC: [PUBLIC]; ERTLOC [PUBLIC] : WORD;
PROCEDURE INIFC [PUBLIC]; BEGIN END;
PROCEDURE ENCYC [PUBLIC]; BEGIN END;
PROCEDURE GTLOC [LEN: WORD; EARG: ACSEW] [PUBLIC]; BEGIN END;
FUNCTION GTLOC [LEN: WORD; EARG: ACSEW] : WORD [PUBLIC];
  BEGIN GTLOC := 0; END;
PROCEDURE PLYC [PUBLIC]; BEGIN END;
PROCEDURE ENCYC [PUBLIC]; BEGIN END;
PROCEDURE INIFC [PUBLIC]; BEGIN END;
```

PROCEDURE TMPCLC [PUBLIC];

```
CONST
  AMULT = 1.0E-03; (* MULTIPLIERS FOR THE THERMISTOR *)
  BMULT = 1.0E-04; (* CONSTANT PARTIALSAS. *)
  CMULT = 1.0E-07;

  ACOEF1 = 0.746597E-03; (* CONSTANTS FOR THE ANALOG PCB TEMPERATURE *)
  BCOEF1 = -0.94521E-02; (* CORRECTION. THESE ARE FOR SERIAL# 001 *)
  CCOEF1 = 0.37104E-04;
  DCOEF1 = -0.40627E-13;
```

VAR

```
A1CONST [ORIGIN 167700] : SINT; (* A-CONSTANT, DIGIT 1 *)
A2CONST [ORIGIN 167701] : SINT; (* A-CONSTANT, DIGIT 2 *)
A3CONST [ORIGIN 167702] : SINT; (* A-CONSTANT, DIGIT 3 *)
A4CONST [ORIGIN 167703] : SINT; (* A-CONSTANT, DIGIT 4 *)
A5CONST [ORIGIN 167704] : SINT; (* A-CONSTANT, DIGIT 5 *)
A6CONST [ORIGIN 167705] : SINT; (* A-CONSTANT, DIGIT 6 *)
```

```
A7CONST [ORIGIN 167706] : SINT; (* A-CONSTANT, DIGIT 7 *)
A8CONST [ORIGIN 167707] : SINT; (* A-CONSTANT, DIGIT 8 *)
A9CONST [ORIGIN 167708] : SINT; (* A-CONSTANT, DIGIT 9 *)
A10CONST [ORIGIN 167709] : SINT; (* A-CONSTANT, DIGIT 10 *)
A11CONST [ORIGIN 167710] : SINT; (* A-CONSTANT, DIGIT 11 *)
```

```
B1CONST [ORIGIN 167711] : SINT; (* B-CONSTANT, DIGIT 1 *)
B2CONST [ORIGIN 167712] : SINT; (* B-CONSTANT, DIGIT 2 *)
B3CONST [ORIGIN 167713] : SINT; (* B-CONSTANT, DIGIT 3 *)
B4CONST [ORIGIN 167714] : SINT; (* B-CONSTANT, DIGIT 4 *)
B5CONST [ORIGIN 167715] : SINT; (* B-CONSTANT, DIGIT 5 *)
B6CONST [ORIGIN 167716] : SINT; (* B-CONSTANT, DIGIT 6 *)
B7CONST [ORIGIN 167717] : SINT; (* B-CONSTANT, DIGIT 7 *)
B8CONST [ORIGIN 167718] : SINT; (* B-CONSTANT, DIGIT 8 *)
B9CONST [ORIGIN 167719] : SINT; (* B-CONSTANT, DIGIT 9 *)
B10CONST [ORIGIN 167720] : SINT; (* B-CONSTANT, DIGIT 10 *)
B11CONST [ORIGIN 167721] : SINT; (* B-CONSTANT, DIGIT 11 *)
```

```
C1CONST [ORIGIN 167722] : SINT; (* C-CONSTANT, DIGIT 1 *)
C2CONST [ORIGIN 167723] : SINT; (* C-CONSTANT, DIGIT 2 *)
C3CONST [ORIGIN 167724] : SINT; (* C-CONSTANT, DIGIT 3 *)
C4CONST [ORIGIN 167725] : SINT; (* C-CONSTANT, DIGIT 4 *)
C5CONST [ORIGIN 167726] : SINT; (* C-CONSTANT, DIGIT 5 *)
C6CONST [ORIGIN 167727] : SINT; (* C-CONSTANT, DIGIT 6 *)
C7CONST [ORIGIN 167728] : SINT; (* C-CONSTANT, DIGIT 7 *)
C8CONST [ORIGIN 167729] : SINT; (* C-CONSTANT, DIGIT 8 *)
C9CONST [ORIGIN 167730] : SINT; (* C-CONSTANT, DIGIT 9 *)
C10CONST [ORIGIN 167731] : SINT; (* C-CONSTANT, DIGIT 10 *)
C11CONST [ORIGIN 167732] : SINT; (* C-CONSTANT, DIGIT 11 *)
```

```
OFFSET [ORIGIN 167733] : REAL; (* ZERO OFFSET FOR THE SYSTEM *)
TEMPLO [ORIGIN 167734] : SINT; (* LOWER TWO DIGITS OF RESULTING TEMPERATURE *)
TEMPHI [ORIGIN 167735] : SINT; (* UPPER TWO DIGITS OF RESULTING TEMPERATURE *)
TEMPSGN [ORIGIN 167736] : SINT; (* SIGN OF TEMPERATURE, 1=NEG C=POS *)

NUMBER [ORIGIN 167737] : INTEGER; (* 16-BIT RESISTANCE PASSED FROM THE PLM MAIN-MODULE *)
AMPAIN [ORIGIN 167738] : INTEGER; (* GAIN SETTING OF PROGRAMMABLE OP-AMP *)
```

```
BUILC, TEPP, X, Y, RES, VFS : REAL; (* LOCAL VARIABLES *)
C1, MVTRUE, MVACC, RANGE : REAL;
OFFSET1, OFFSET2 : REAL;
ACOEF1, BCOEF1, CCOEF1, DCOEF1 : REAL; (* COEFFICIENTS FOR A/D SPAN DRIFT DUE TO TEMP. *)
ZROFLAG [ORIGIN 167739] : SINT; (* 0=USE STORED OFFSET, 1=CALC. & STORE NEW OFFSET *)
ACONST [ORIGIN 167740] : REAL; (* RESULTANT A CONSTANT *)
BCONST [ORIGIN 167741] : REAL; (* RESULTANT B CONSTANT *)
CCONST [ORIGIN 167742] : REAL; (* RESULTANT C CONSTANT *)
ABSTP [ORIGIN 167743] : INTEGER; (* READING FROM THE CN-CARD TEMPERATURE SENSOR *)
```

```
BEGIN
  IF ACCNST=C.C THEN
    BEGIN
```

```
      BUILO := 0.0;
      BUILO := A1CONST + (A2CONST / 10.0) + (A3CONST / 100.0) + (A4CONST / 1000.0) + (A5CONST /
```

```

BUILD := BUILD + (A6CONST / 1000000.0) + (A7CONST / 1000000.0) + (A8CONST / 1000000.0) +
      (A9CONST / 10000000.0);
BUILD := BUILD + (A10CONST / 100000000.0) + (A11CONST / 100000000.0);
ACONST := BUILD * AMULT; (* FINAL RESULT *)
END;

IF BCCNST=0.0 THEN
  BEGIN
    BUILD := 0.0;
    BUILD := B1CONST + (B2CONST / 10.0) + (B3CONST / 100.0) + (B4CONST / 1000.0) + (B5CONST /
      10000.0);
    BUILD := BUILD + (B6CONST / 100000.0) + (B7CONST / 1000000.0) + (B8CONST / 10000000.0) +
      (B9CONST / 100000000.0);
    BUILD := BUILD + (B10CONST / 1000000000.0) + (B11CONST / 10000000000.0);
    BCONST := BUILD * BMULT; (* FINAL RESULT *)
  END;

IF CCCNST=0.0 THEN
  BEGIN
    BUILD := 0.0;
    BUILD := C1CONST + (C2CONST / 10.0) + (C3CONST / 100.0) + (C4CONST / 1000.0) + (C5CONST /
      10000.0);
    BUILD := BUILD + (C6CONST / 100000.0) + (C7CONST / 1000000.0) + (C8CONST / 10000000.0) +
      (C9CONST / 100000000.0);
    BUILD := BUILD + (C10CONST / 1000000000.0) + (C11CONST / 10000000000.0);
    CCONST := BUILD * CMULT; (* FINAL RESULT *)
  END;

CASE AMPGAIN OF
  C:BEGIN
    RANGE := 5.0; (* 5.0 VOLT RANGE *)
    CFFSETM := 7.561E-03; (* THIS RANGE IS UNUSED *)
    CFFSETB := -13.89; (* NUMBER OF BITS DRIFT IN A/D OFFSET BETWEEN -4C & +20 *)
    ACOEF1 := 5.25372;
    CCOEF1 := 0.75804E-04;
    CCOEF2 := 0.18373E-06;
    CCOEF3 := 0.40731E-10;
  END;

  1:BEGIN
    RANGE := 2.5; (* 2.5 VOLT RANGE *)
    CFFSETM := 7.561E-03; (* THIS RANGE IS UNUSED *)
    CFFSETB := -14.0; (* NUMBER OF BITS DRIFT IN A/D OFFSET BETWEEN -4C & +20 *)
    ACOEF1 := 2.43999;
    CCOEF1 := -0.37509E-03;
    CCOEF2 := -0.24893E-07;
    CCOEF3 := -0.13397E-11;
  END;

  2:BEGIN
    RANGE := 1.0; (* 1.0 VOLT RANGE *)
    CFFSETM := 7.561E-03; (* NUMBER OF BITS DRIFT IN A/D OFFSET BETWEEN -4C & +20 *)
    CFFSETB := -14.69;
    ACOEF1 := 1.12125;
    CCOEF1 := -0.01403E-04;
  END;

  CCOEF1 := 0.91495E-03;
  CCOEF2 := 0.20402E-11;
END;

  3:BEGIN
    RANGE := 0.5; (* 0.5 VOLT RANGE *)
    CFFSETM := 7.561E-03; (* NUMBER OF BITS DRIFT IN A/D OFFSET BETWEEN -4C & +20 *)
    CFFSETB := -15.38;
    ACOEF1 := 0.56223;
    CCOEF1 := -0.44628E-04;
    CCOEF2 := -0.24900E-07;
    CCOEF3 := 0.55090E-11;
  END;

  4:BEGIN
    RANGE := 0.25; (* 0.25 VOLT RANGE *)
    CFFSETM := 7.561E-03; (* NUMBER OF BITS DRIFT IN A/D OFFSET BETWEEN -4C & +20 *)
    CFFSETB := -16.89;
    ACOEF1 := 0.2810247;
    CCOEF1 := -0.40381E-03;
    CCOEF2 := -0.49134E-07;
    CCOEF3 := -0.48348E-11;
  END;

  5:BEGIN
    RANGE := 0.10; (* 0.10 VOLT RANGE *)
    CFFSETM := 7.561E-03; (* NUMBER OF BITS DRIFT IN A/D OFFSET BETWEEN -4C & +20 *)
    CFFSETB := -18.19;
    ACOEF1 := 0.14505;
    CCOEF1 := -0.44321E-04;
    CCOEF2 := -0.48041E-08;
    CCOEF3 := -0.21771E-12;
  END;

END;

VFS := ACOEF1 + (BCOEF1 * AMBTP) + (CCOEF1 * AMETMP * AMBTP) + (DCOEF1 * AMBTP * AMBTP * AMBTP); (* ACTUAL
FULL SCALE VCLTAGE *)

Y := NUMBER + (OFFSETH * AMBTP + OFFSETB); (* CALC. THE OFFSET DRIFT FOR THE A/D & CORRECT "NUMBER" *)
MVADC := (Y/4095)*VFS; (* CALC. THE ACTUAL VOLTAGE GOING INTO THE A/D CONVERTER *)

C1 := ACOEF2 + (BCOEF2 * AMBTP) + (CCOEF2 * AMBTP * AMBTP) + (DCOEF2 * AMBTP * AMBTP * AMBTP); (* ANALOG PCB CORRECTION FACTOR FOLLOWS... *)
MVTRUE := MVADC + C1; (* CALC. THE ACTUAL mV ACROSS THE THERMISTOR *)

RES := MVTRUE / 12.207E-06; (* FINALLY CONVERT mV TO RESISTANCE *)

Y := LN(RES);
X := ACONST + BCCNST * Y + CCCNST * Y * Y + SOR(Y); (* STEINPART, HART EQUATION *)
TEMP := 1.0 / X - 273.15; (* CONVERT TO CELSIUS *)

IF ZROFLAG = 1 THEN
  BEGIN
    CFFSET := TEMP - 0.01; (* CALC. OFFSET FOR THERPISTOR IN TRIPLE-POINT CELL *)
  END;

```

```

                                ZROFLAG := 0;          (* RESET THE FLAG *)
                                END;
                                TEMP := TEMP - OFFSET;      (* REMOVE THE SYSTEM OFFSET *)
                                IF TEMP >= 0.0 THEN
                                    TEMPSGN := 0;          (* POSITIVE TEMPERATURE *)
                                ELSE
                                    TEMPSGN := 1;          (* NEGATIVE TEMPERATURE *)
                                TEMP := ABS(TEMP);          (* CHANGE TO POSITIVE VALUE.. *)
                                TEMP := TEMP * 0.005;      (* ROUND TO 2 DECIMAL PLACES *)
                                TEMPH1 := TRUNC(TEMP);      (* UPPER TWO DIGITS OF TEMPERATURE *)
                                TEMP := TEMP - TEMPH1;      (* KEEP THE FRACTIONAL PART *)
                                TEMPLC := TRUNC(TEMP * 100.0); (* LOWER TWO DIGITS OF TEMPERATURE *)
                                END;
                                (* END OF TEMPCALC. PROCEDURE *)
                                BEGIN END.              (* BEGIN & END OF MAIN MODULE *)

```

```

                                >>> RAM.VAC <<<
N8080
PUBLIC ACORCC, AC2RCC, AC3RCC, AC4RCC, REGHCC
PUBLIC CLNECC, CSX2CC, CUPHCC, ECNJCC, EFGXCC
PUBLIC ENDMCC, FACRCC, FRPXCC, GAPHCC, TOSFCC
PUBLIC HCRVCC, INTNCC, J10RCC, J20RCC, J30RCC
PUBLIC J40RCC, LCNJCC, POSRCC, PS2RCC, PNUXCC
PUBLIC REFCCC, REFC2CC, REFC3CC, REFC4CC, RXCC
PUBLIC SP5LCC, STK6CC, STK4CC, STK2CC, XX10CC, XX2GCC
PUBLIC XX3GCC, XX4GCC, IMPRY

```

dseg

```

FRPXCC:DS1
ACORCC:DS1
FACRCC:DS1
AC2RCC:DS1
AC3RCC:DS1
AC4RCC:DS1
REGHCC:DS1
CLNECC:DS1
CSX2CC:DS1
CUPHCC:DS1
ECNJCC:DS1
ENDMCC:DS1
HCRVCC:DS1
J10RCC:DS1
J20RCC:DS1
J30RCC:DS1
J40RCC:DS1
POSRCC:DS1
PS2RCC:DS1
PNUXCC:DS1
REFCC:DS1
REFC2CC:DS1
REFC3CC:DS1
REFC4CC:DS1
RXCC:DS1
STK6CC:DS1
STK4CC:DS1
STK2CC:DS1
XX10CC:DS1
XX2GCC:DS1
XX3GCC:DS1
XX4GCC:DS1

```

ECNJCC:DS1

CSEG
LCNJCC:DEECNJCC-FRFXC2

BHMY: DM C 190CM ; fill in with lowest AVAILABLE ram location

DSEG
EFGXCC:DS1
CSEG
GAPHCC:DM128

```

CSEG
HCRVCC:DS1
INTNCC:DS1
J10RCC:DS1
J20RCC:DS1
J30RCC:DS1
J40RCC:DS1
PNUXCC:DS1
REFC2CC:DS1
REFC3CC:DS1
REFC4CC:DS1
END

```

NR06C

>>> SXENTX.MAC <<<

TITLE Entx - entry / exit code for Microsoft #020 Pascal
SUBTTL Sterplex-II version

System and procedure entry and exit point routines

William Fox, April 1992

Globals

PUBLIC \$START
PUBLIC EEXGCC, BTEGCC, ENDCGCC, ERTEGCC, FXSHCC

EXTRN BEGHCQ, BEGQCC, CSXECQ, CURHCQ, EFGXCC
EXTRN ENDMCC, ENDCGCC, ENDCGCC, ENDCGCC, ENDCGCC
EXTRN FRPXCQ, GAPHCQ, HDRFCQ, HDRVCQ, INIUCQ
EXTRN INTHCC, PNUXCQ, RESECC, RESECC, RESECC
EXTRN RESECC, STKBCQ, STKBCQ

EXTRN \$MMRY

Defined symbols

MEMIP SET 1FFFFH ;Top of available memory (note constants are octal)
EXIT SET C006H ;addr of exit routine

\$START: JMP BEGXCQ

SUBTTL Begxcq - system entry point and initialization routine

Main system entry point and initialization routine

- 1) Init the stack pointer, frame pointer, and stkhq.
- 2) Init EFGXCQ to zero.
- 3) Init the heap unit, ie. set beghcq, curhcq, endhcq, and stkhq.
- 4) Init the machine error context by zeroing resecc.
- 5) Init the source error context by zeroing csxecq.
- 6) Init the unit initialization system by zeroing pnuxcc.
- 7) Init the file system by zeroing hdrfcq and hdrvcq.
- 8) Init the machine dependent file system by calling iniucq.
- 9) Call the escape initialization routine begqcc.
- 10) Call the main program at entgq.

BEGXCQ: POP C ;(GMT) return address of PLM code
LXI P, 0+C
SHLD CSXECQ ;init the frame pointer
SHLD FRPXCQ
SHLD PNUXCQ
SHLD RESECC
SHLD HDRFCQ

SHLD HDRVCQ
LXI P, ((PPTF-2) AND 1777700)+C;HL := highest address -2 on even
;boundary
SPHL ;init the stack pointer
PUSH E ;(GMT) put return address of PLM code on the new stack
DCX P
DCX P
SHLD STKBCQ ;upon stack-heap collision, restore the sp from this
; also used by rfpqcc for long jumps

EFGXCQ is used by ENDCGCC as a 'have I been here before' flag, to avoid recursive errors with ENDCGCC.

XRA A
STA EFGXCQ

Init the heap unit

LHLD \$MMRY
INX P ;round it to even
MOV B, L
ANI 1760
MOV L, A
MVI P, 0
;mark the first block as free
SHLD BEGHCQ
SHLD CURHCQ
SHLD ENDMCC
CALL FXSHCC

CALL INIUCQ ;init the machine dependent file system
RET ;(GMT) return to the PLM main program
CALL BEGQCC ;allow the user to init things
CALL ENTGCC ;call the main program
JMP ENDCGCC ;all finished
SUBTTL Erteq - entry point helper for runtime

procedure brteq (offset: word)

\$Runtime entry point helper - if this is the top level entry into the runtime system, store the user routine's error context info for esseq.

DB C

BRTECC: POP M ;HL := my return address
XCHG ;HL := offset to my caller's return address
LDA RESECC ;DE := offset to my caller's return address
MOV R, A ;have I already done this?
LDA RESECC+1
ORA E
JNZ BRTX ;yes

```

MOV      A,C                      ;DE := - of offset to my caller's return address
CHA      C,A
MOV      A,E
CHA      E,A
MOV      C,FRPXCO
INX      C                      ;HL := my caller's frame pointer
LHLD     C,E,M                  ;HL := address of my caller's return address
MOV      E,M
INX      E,M
MOV      C,M
INX      C
XCHG     REPECC                 ;store the user's pc
SHLD     E,M                    ;DE := my caller's saved frame pointer
XCHG     E,M
MOV      C,M
XCHG     REPECC                 ;save the user's frame pointer
SHLD     REPECC                 ;the PCBO is an unsegmented machine...
LXI      PC,0                  ;so save 0 as the segment base
SHLD     REPECC                 ;my caller's caller's stack pointer is equal
LHLD     C,2+C                 ;to my caller's frame pointer + 2
DAD      C
SHLD     REPECC
SRTX:    RET                      ;end of life compil

SUBTTL   Endxcq - system exit point and termination routine

Main system termination routine
1) If EFGXCQ is zero, invoke the escape termination routine ENDCCC.
2) Close all open files by calling endycq.
3) Invoke the machine dependent file system termination routine, enduqc.
Return to Sterlex by a jump to EXIT

ENDXCQ:   DB      C
          LDA      EFGXCQ
          CRA      A
          JNZ      ENX1
          INR      A
          STA      EFGXCQ
          CALL     ENDOC
          CALL     ENDYC
          CALL     ENDUC
          JMP      EXIT
          SUBTTL   Ertecq - exit point helper for $runtime

procedure ertecq;
$Runtime exit point helper - if I'm returning from the top level
entry into the runtime system, restore the error context

```

```

DB      C

ERTECC:   LHLD     RESEQC         ;HL := error context sp
          XCHG     C,DE           ;DE := c(reseqc)
          LHLD     FRPXCO        ;HL := current fp
          INX      C             ;HL := my caller's caller's sp
          INX      C
          MOV      A,L           ;if sp greater than or equal to reseqc,
          SUB      E,A           ; zero error context
          MOV      A,M
          SBB      C
          JC       ERTX
          LXI      PC,C+C
          SHLD     RESEQC
          RET

ERTX:     SUBTTL   Fxshq - public subroutine to compute stkhaq
          ;end of life compil

STKMCQ is used to detect collision (within GAPHCQ) of the stack and
the heap: if the current sp + STKMCQ does not produce a carry, they're
too close. STKMCQ should always equal - (ENDMCQ + GAPHCQ).

FXSMCC:   LHLD     ENDMCQ
          XCHG     C,DE           ;DE := endhqa, HL := scratch
          LHLD     GAPHCQ
          DAD      C             ;HL := endhqa + caohcc
          MOV      A,L           ;negate it
          CHA      A,L
          MOV      L,A
          MOV      H,M
          CMA      A
          MOV      A,A
          INX      C             ;make it two's complement
          SHLD     STKMCQ
          MVI      A,3770
          STA      INTMCQ
          RET
          END      $START

;
          END

;release the heap pect

```

APPENDIX B: INSTRUCTIONS FOR OPERATION OF THERMISTOR METER VERSION 1.0

Getting started

1. Connect a thermistor to the connector on the top panel. Make sure all connections are clean and dry. If the instrument is submerged while in use, remove the thermistor connector and make sure that the pins are dry. Any water could cause a leakage path around the thermistor and have the effect of raising the measured temperatures slightly. This effect will be more pronounced when low temperatures are being measured.
2. Turn the CHARGE/OFF/ON switch to the ON position.
3. If the constants for this thermistor have been entered, the actual temperatures will be displayed. If different constants are required then follow the Entering constants procedure. If "FFFF" is on the display then follow the Entering constants procedure from step 2. (You will not be allowed to exit the FUNCTION mode until all three constants have been entered.)

Entering constants

1. Enter "F" to invoke the FUNCTION mode. "FFFF" should be on the display.
2. Enter "EA,", "EB" or "EC" to enter a new A, B or C constant respectively. For example to enter a new "B" constant simply type "EB."
3. The display will now show "AAAA," "BBBB" or "CCCC." The 11 digits must now be entered, with trailing zeroes if needed. The display will scroll to the left as they are entered. If you make a mistake you may back up a digit at a time by entering "B". This will cause the display to scroll to the right. Any digits that roll off the right end of the display are lost and must be reentered. After entering the eleventh digit you will be returned automatically to function mode signified by "FFFF" on the display.
4. Repeat steps 2 and 3 as needed.
5. The constants may be verified by following the procedure Displaying constants from step 2.
6. Return to the temperature mode by entering "EE."

Displaying constants

1. ENTER "F" to get into the FUNCTION mode. "FFFF" should be on the display now.
2. Enter "DA," "DB" or "DC" to display the A, B or C constant. The first four digits of the mantissa will be displayed. You may look at the rest of the digits by pressing any key. Each keypress will bring up the next group of four digits until all 12 have been seen. The third keypress will bring you back to the function mode, signified by "FFFF" on the display.
3. Return to the temperature mode by entering "EE."

NOTE: If you make a mistake and enter an illegal command, the display will blink "EEEE" at you briefly then return to "FFFF". You may now try again.

There are rechargeable batteries in the meter. They may be charged in the following manner:

1. Turn the CHARGE/OFF/ON switch to the OFF position.
 2. Locate the line cord. It has a three-pin grounded plug on one end and an eight-pin Bendix connector on the other.
 3. Mate this Bendix connector to the one on top of the instrument and lock it in place. Make sure the pins are clean and dry.
 4. Plug the other end of the line cord into a grounded 110-Vac outlet.
 5. Turn the CHARGE/OFF/ON switch to the CHARGE position.
- The batteries should be fully charged in about 8 hours. The meter should run continuously for 6-8 hours under normal conditions. The battery life may be less when the ambient temperature is below -20°C.

CAUTION: The meter has been sealed completely to waterproof it. Removal of the top panel or turning of any screw heads on the outside of the box will break this seal.

END

Dtic

5-86