Microcontroller-based data acquisition system.

Adrian Chepetan

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/2726
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®
NOTE TO USERS

This reproduction is the best copy available.

UMI
MICROCONTROLLER BASED DATA ACQUISITION SYSTEM

By

Chepehan Adrian

A Thesis
Submitted to the Graduate Studies and Research Through the Department of Electrical Engineering in Partial Fulfillment Of the Requirements for the Degree of Master of Applied Science at The University of Windsor

January 19, 2001
Windsor, Ontario, Canada
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.


**ABSTRACT**

The Aluminum manufacturing represents a major segment of the today Industries. The body of the engines is manufactured using Aluminum Alloy. The Thermal Analysis involves the measurement of temperature and time in order to produce cooling curves to describe the properties of a specific test sample. The cooling curves are the meaning to establish the proper composition of the Alloy.

The Data Acquisition System represents the meaning with which the analysis is performed. On the marketplace already exist many Data Acquisition Systems dedicated for this task. One of them is currently used in the Aluminum Alloy Analysis Research at the University of Windsor. This is a National Instrument System special configured to acquire Thermocouple signals.

To perform an accurate Data Acquisition the most important criteria is ‘Accuracy’. One way to achieve this is to fully understand the entire System involved in this process. Designing a System is the best way to gain this understanding.

It was the objective of this Thesis to develop a very less cost system with similar characteristics than the National Instruments System. The Design for this System was performed and all the necessary programming was done to achieve the desired performance. The testing carried out over the system shows positive aspects conform the expectations.
In the same time there where observed possible improvements, which will, may increase the performance of the actual prototype. Therefore this project represents a first step. The data accumulated in the time of the design and all the results from the tests performed over the system may constitute the reference for a further development in this direction.
In the memory of my father, Pavel D. Chepetan
ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor Dr. James J. Soltis for his guidance and his support throughout the entire program, Dr. Jerry H. Sokolowsky and his entire Team involved in the Aluminum Alloy Research, for the pertinent advice and encouragement while developing this project.

I would also like to thank Dr. R.G.Gaspar and Dr. X.Chen for their useful suggestions throughout this project.

A special thanks to my family for the entire support and understanding.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>DEDICATION</td>
<td>v</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>vi</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>vii</td>
</tr>
<tr>
<td>CHAPTER 1</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Aluminum Composition Analysis</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Instruments and Procedures</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Thermocouple Principle</td>
<td>4</td>
</tr>
<tr>
<td>1.3.1 Thermocouple Type K (Chromel/Alumel)</td>
<td>5</td>
</tr>
<tr>
<td>1.4 Precautions and Considerations for Using Thermocouples</td>
<td>7</td>
</tr>
<tr>
<td>1.4.1 Connection Problems</td>
<td>7</td>
</tr>
<tr>
<td>1.4.2 Lead Resistance</td>
<td>7</td>
</tr>
<tr>
<td>1.4.3 Decalibration</td>
<td>7</td>
</tr>
<tr>
<td>1.4.4 Noise</td>
<td>8</td>
</tr>
<tr>
<td>1.4.5 Common Mode Voltage</td>
<td>8</td>
</tr>
<tr>
<td>1.4.6 Thermal Shunting</td>
<td>9</td>
</tr>
<tr>
<td>CHAPTER 2</td>
<td>10</td>
</tr>
<tr>
<td>2.1 System Description</td>
<td>10</td>
</tr>
<tr>
<td>2.1.1 Specification of Analog-to-Digital Converters</td>
<td>10</td>
</tr>
<tr>
<td>2.1.2 Resolution</td>
<td>10</td>
</tr>
<tr>
<td>2.1.3 Integral Linearity</td>
<td>11</td>
</tr>
<tr>
<td>2.1.4 Gain Error</td>
<td>11</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

1.1 The most usual thermocouple and the most important characteristics ........................................4  
1.2 Mode Register Configuration ........................................................................................................26  
1.3 Operating Mode for the AD7731 ..................................................................................................27  
3.3 Input Range Selection of the AD7731 ..........................................................................................29  
3.4 Channel Selection of AD7731 .......................................................................................................30  
4.1 Settling Time and Throughput Rate for all the possible modes (X=1 or 0) ..................46  
4.2 Output Noise vs. Input Range and Update Rate (CHP = 0, SKIP = 1)  
   Typical Output RMS Noise in μV .................................................................................................47  
4.3 Peak-to-Peak Resolution vs. Input Range and Update Rate  
   (CHP = 0, SKIP =1) Peak-to-Peak Resolution in Bits .................................................................48  
4.4 Output Noise vs. Input Range and Update Rate (CHP = 1, SKIP = 0)  
   Typical Output RMS Noise in nV .................................................................................................50  
4.5 Peak-to-Peak Resolution vs. Input Range and Update Rate  
   (CHP = 1, SKIP = 0) ....................................................................................................................53  
5.1 Selection for the BAUD rate of the SCI communication .........................................................58  
5.2 SPI clock rate selection .................................................................................................................58
CHAPTER 1

1.1 Aluminum composition analysis

In the process of manufacturing of Aluminum alloy one major problem is to establish the proper composition. The spectral analysis of the composition of the alloy is an expensive and time-consuming procedure involving specialized instruments. For this particular problem of casting blocks for the engines, the composition of aluminum is determinant for the quality of the final product. Many experiments were conducted in the past to find a proper method to investigate the composition of melted alloy in the process of elaboration.

One inexpensive method involves measuring the temperature of solidification of an alloy probe between 800 and 350 degrees Celsius, and recording the curve of solidification, then further compared with a standard curve in order to establish the composition of the alloy. The research is done at the University of Windsor by a team conducted by dr. Jerry Sokolowsky.

During time of solidification, the temperature of the sample (melted alloy) is monitored and the curve of solidification is observed. It was established that for a certain percent of the composition of the alloy, a specific curve of solidification would result. The harmful impurities, which could be mixed in the alloy, will disturb the curve of solidification from the Desired Pattern. This Pattern represents a curve of solidification of an alloy considered without impurities and therefore the one acceptable as a final product. In order to draw a curb of an alloy, the temperature of solidification is monitored by a data acquisition system, which use a thermocouple as a temperature transducer.
1.2 Instruments and Procedures

The setup for this test includes a National Instrument Data Acquisition Board connected to a PC under the LabView environment. A standard National Instrument rack is holding all the cards used for the application.

The first card is a multifunctional, multi channel SCXI-1120 (8 programmable amplifiers) combined with an SCXI-1124 6 channel analog digital converter. The link between these boards is provided via an SCXI-bus configured on the back plate of the rack.

The next card, which is connected to the thermocouple, is an SCXI-1100 module, which features an isothermal enclosure with a precise transducer to measure the point of connection for the cold junction correction. In addition to these boards the system contains a digital I/O board and an SCXI-1301 terminal block to interface to the sensors.

The software, ‘LabView’ allows all the system parts to be configured to achieve the desired setup for the specific task needed. This system has certain advantages in particular because of the accuracy of the results due to the enhanced features of the boards. On the other hand this setup is very expensive and too enhanced for the purpose of this test. Another weakness of this board is that the components used in the original design have less performance than the components available at this time on the market.

In this project, it was designed an efficient portable tool, using digital chips available at this time on the market, at very low cost and high performance. The “switch board” of the system is a Motorola 68HC12, 16 bits microcontroller which embedded the function of traffic cop for the “measured signal” from the thermocouple and the “configuring signal” necessary to program the parameters of the system. The microcontroller is connected to a host PC via a serial communication interface SCI capable of handling communication with a maximum of 38400 baud. On the other end, the microcontroller is connected via a high-speed synchronous serial peripheral interface (SPI) capable of communicating with maximum 4000 Kbits/sec with the A/D converter.
In the same time the microcontroller is used to buffer the data received from the A/D converter and also store the default parameters of the system.

The A/D converter is a high performance 24 bit Sigma-Delta modulator (Analog Devices, AD7731), with programmable parameters (output rate, bandwidth and notch filter, gain, resolution etc). The parameters of the A/D converter are tuned to maximize the immunity to the noise of the system.

The trade off between the output rate and the rejection of the noise will allow an 800Hz rate with no more than 2 microvolts noise figure for a level of the maximum 80 milivolts thermocouple signal. The thermocouple is delivering 40 microvolts per degree Celsius, or for a range of temperature from 350 to 800 degree Celsius, therefore the signal will range between 14 to 30 mV.

To increase the signal/noise ratio the preamplifier built in the input stage of the part was used. In order to minimize the noise from the thermocouple the proper shielding and grounding of the signal was performed.

The cold junction problem is solved by using a precision temperature transducer (AD592AN) mounted on the same point with the thermocouple junction to the system. The cold junction compensation is performed using the calibration feature of the A/D converter. The specific problems related to the conditioning of the signal from the thermocouple to the analog digital converter will be discussed in a special paragraph dedicated for this problem. The acquired signal from the thermocouple after is converted and conditioned in the system is transferred to the host PC under a program monitor written in Visual Basic. This also offers the front-end feature of the programming the parameters of the system.

Data stored in the PC is available to be imported to the Simulink environment where it may to be processed with the DSP toolbox. One of the main advantages of using Simulink “processing” is the possibility of identifying time invariant noise signals, which are specific to the environment where the acquisition process is taking place. As soon these components are identified it will be possible to program the system to reject them.
1.3 Thermocouples principle

In 1822, Thomas Seebeck discovered that the junction between two metals generates a voltage, which is a function of temperature. Thermocouples are based on this Seebeck effect. Although almost any two types of metal can be used to make a thermocouple, a number of standard types are used because they possess predictable output voltages and large temperature gradients. Thermocouples are the most popular temperature sensors. They are cheap, interchangeable, have standard connectors and can measure a wide range of temperatures. The main limitation is accuracy, system errors of less than 1°C can be difficult to achieve. See Table No.1.1 for the most used thermocouples.

<table>
<thead>
<tr>
<th>Type (ANSI CODE)</th>
<th>ALLOY CONSIDERATION</th>
<th>TEMPERATURE</th>
<th>OUTPUT RANGE mV</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Platinum/Rhodium</td>
<td>0 °C to 1700 °C</td>
<td>0 to 12.426</td>
</tr>
<tr>
<td>E</td>
<td>Chromel/Constantan</td>
<td>-200°C to 900°C</td>
<td>-8.824 to +68.783</td>
</tr>
<tr>
<td>J</td>
<td>Iron/Constantan</td>
<td>0 °C to +750°C</td>
<td>0 to +42.283</td>
</tr>
<tr>
<td>K</td>
<td>Chromel/Alumel</td>
<td>-200°C to 1250°C</td>
<td>-5.973 to +50.633</td>
</tr>
<tr>
<td>N</td>
<td>Nicrosil/Nisil</td>
<td>-270°C to 1300°C</td>
<td>-4.345 to +47.502</td>
</tr>
<tr>
<td>R</td>
<td>Platinum/Rhodium</td>
<td>0°C to +1450°C</td>
<td>0 to +16.741</td>
</tr>
<tr>
<td>S</td>
<td>Platinum/Rhodium</td>
<td>0°C to +1450°C</td>
<td>0 to +14.973</td>
</tr>
<tr>
<td>T</td>
<td>Copper/Constantan</td>
<td>-200°C to +350°C</td>
<td>-5.602 to +17.816</td>
</tr>
</tbody>
</table>

Table. No. 1.1

The most usual thermocouple and the most important characteristics
1.3.1 Thermocouple Type K (Chromel / Alumel)

Type K is the general-purpose thermocouple. It is low cost and it is available in a wide variety of probes. Thermocouples are available in the -200°C to +1200°C range. Sensitivity is approx 41µV/°C. Standard tables show the voltage produced by thermocouples at any given temperature for example in the above diagram, the K type thermocouple at 300°C will produce 12.2mV. Unfortunately it is not possible to simply connect up a voltmeter to the thermocouple to measure this voltage, because the connection of the voltmeter leads will make a second, undesired thermocouple junction.

Figure No1.1 shows the principle of the cold junction [3]. To make accurate measurements, this must be compensated using a technique known as cold junction compensation (CJC). All standard thermocouple signals allow for this second thermocouple junction by assuming that it is kept at exactly zero degrees centigrade.

In the laboratory environment this is done with a carefully constructed ice bath (hence the term 'cold' junction compensation). Maintaining an ice bath is not practical for most industrial measurement applications, so instead the actual temperature at the point of connection of the thermocouple wires to the measuring instrument is monitored.

Typically cold junction temperature is sensed by a precision thermistor with good thermal contact to the spot where is the junction to the input connectors of the measuring system. This second temperature reading is added with the polarity reversed to the reading from the thermocouple itself subtracting the unneeded voltage from the real one. For less critical applications, a semiconductor temperature sensor performs the cold junction compensation. By combining the signal from this semiconductor with the signal from the thermocouple, the correct reading can be obtained without the need or expense to record two temperatures.

Understanding of cold junction compensation is important; any error in the measurement of cold junction temperature will lead to the same error in the measured temperature from the thermocouple tip.
Figure No1.1

Principle of Cold Junction

Figure No1.2

Cold Junction Compensation
1.4 Precautions and Considerations for Using Thermocouples

Most measurement problems and errors with thermocouples are due to an incorrect setup for connecting the thermocouple to the system. Clarification for some of the aspects related with these problems will be done further.

1.4.1 Connection problems.

Unintended thermocouple junctions appear between the existing thermocouple cable wires and the attached measurement cable. It is known that any junction of two different metals will cause a voltage. If there is a need to increase the length of the leads from the thermocouple, then the correct type of thermocouple extension wire should be used. For our project a type K wire will be needed for type K thermocouples. Using any other type of wire will introduce a thermocouple junction. Any connectors used must be made of the correct thermocouple material and correct polarity must be observed [3].

1.4.2 Lead Resistance.

To minimize thermal shunting and improve response times, thermocouples are made of thin wire. This can cause the thermocouple to have a high resistance which can make it sensitive to noise and can also cause errors due to the input impedance of the measuring instrument. A typical exposed junction thermocouple with 32AWG wire (0.25mm diameter) will have a resistance of about 12 ohms / meter. If thermocouples with thin leads or long cables are needed, it is worth keeping the thermocouple leads short and then using thermocouple extension wire (which is much to thicker, so has a lower resistance) run between the thermocouple and measuring instrument. The cable used in our setup has a total resistance of 8.6 Ohm.

1.4.3 Decalibration

Decalibration is the process of unintentionally altering composition of thermocouple wire. The usual cause is the oxidation of the metal at the extremes of operating temperature. Another cause is impurities and chemicals from the metal vapors
diffusing into the thermocouple wire. The special coating with ceramic and the entire design of the thermocouple is intended to overcome this problem.

1.4.4 Noise.

The output from a thermocouple is a small signal, so it is vulnerable to electrical noise. The harmful environment where the measurement of the temperature of alloy takes place is a critical factor, which should be, taken into consideration when designing the interface between the thermocouple and the measuring system. The differential input of the measurement amplifier rejects any common mode noise (signals that are the same on both wires) so noise can be minimized by twisting the cable together to help ensure both wires to pick up the same noise signal. Additionally, the design uses an analog-to-digital-converter with chopping and filtering capability, which helps average out any remaining noise. If operating in an extremely noisy environment, (large motor, or SCR converters) it is recommended considering using a screened extension cable. These problems and the design criteria will be discussed further in the amplifier section.

1.4.5 Common Mode Voltage.

Since thermocouple signal are very small, much larger undesirable voltages could exist at the input of the system. These voltages can be caused either by induction or because of a bad grounding procedure. If there are any poor connections to the ground a few volts may exist between the shield or ground plane of the system and the potential earth point. These signals are again common mode (both thermocouple wires are referred to this potential) so it will not cause a problem with the instrument’s input if they are not too large. If this voltage is greater than the capability of rejection of the system to the common mode voltage, then measurement errors will result. Common mode voltages can be minimized using the same cabling precautions outlined for noise, and also by using insulated (floated) thermocouples, and a very good consideration of the ground problem.
1.4.6. Thermal Shunting.

All thermocouples have a certain mass. Heating this mass takes energy so will affect the temperature you are trying to measure. Consider the example of measuring the temperature of liquid alloy in a sample cell: there is a potential problem: the heat energy will travel up the thermocouple wire and dissipate to the atmosphere so reducing the temperature of the alloy around the wires. A similar problem can occur if the thermocouple is not sufficiently immersed in the alloy, due to the cooler ambient air temperature on the wires. Thermal conduction may cause the thermocouple junction to have a different temperature to the alloy itself. In the above example a thermocouple with thinner wires may help, as it will cause a steeper gradient of temperature along the thermocouple wire at the junction between the liquid and ambient air. If thermocouples with thin wires are used, consideration must be paid to lead resistance. The use of a thermocouple with thin wires connected to a much thicker thermocouple extension wire often offers the best compromise The conclusion, which rises from this analysis, will help to determinate the best profile for the thermocouple and the type of interfacing to the system.
CHAPTER 2

2.1 System Description

2.1.1 Specifications of Analogue-to-Digital Converters

Now on the market are available a variety of analog digital converters. The criteria to choose a proper one for this application are based on selecting the specific characteristics necessary to accomplish the desired performance. The most important characteristics are: resolution, linearity, offset errors, acquisition time, throughput, integration time and re-calibration.

2.1.2 Resolution

The resolution of the A-D converter is the number of steps the input range is divided into. The resolution is usually expressed as bits (n) and the number of steps is 2 to the power n. A converter with 8-bit resolution, for instance, divides the range into \(2^8\), or 256, steps. In this case a 0-10 V range will be resolved to 39mV, and a 0-100 mV range will be resolved to 0.39mV. Although the resolution will be increased when the input range is narrowed, there is no point in trying to resolve signals below the noise level of the system: all you will get is unstable readings.

The analog digital converter AD7731 used for this application is a Delta –Sigma modulator type with a 24 bits resolution. However the resolution of the part could be decreased in order to achieve a faster throughput of the signal from the first stage of the system to the microcontroller. The real resolution will be derivate from these characteristics also from the others characteristics like settling time, noise and drift. For a resolution of 16 bits and an input range of the signal between 0 to 80 mV the resolution or the level of the LSB (least significant bit) will be \(80/2^{16} = 1.22\) microvolts.
2.1.3 Integral Linearity

Ideally an A-D converter with n-bit resolution will convert the input range into $2^n - 1$ equal steps (256 steps in the case of a 8-bit, and 65536 steps for 26-bit converter). In practice the steps are not exactly equal, which leads to non-linearity in a plot of A-D output against input voltage. The measure of integral nonlinearity is the deviation from a straight line passing through the ends points of the transfer function see Figure No.2.1 [4] and for AD7731 this is 15 ppm, this means for a scale of 80 mV there are 15x0.08 microvolts = 1.2 microvolts maximum error or 0.028 degree C.

![Figure No. 2.1](image)

Gain and Nonlinearity Errors

2.1.4 Gain error

Due to the increase of the temperature among the part because of the power dissipated by these, the slope of the transfer function will slide from the line passing the zero and full-scale points - see Figure.No.2.1 [4]. This could be observed when the input is maintained to the same value for a period of time and the output will change. This error
for the AD7731 could be corrected due to the calibration system embedded in the part scaling the signal with a value obtained in the time of calibration and stored in the gain register. However before calibration the gain error is 3000 ppm of the full scale and after calibration this will be under 100 ppm of full scale or in value of microvolts this error could be max 100x0.08=8microvolts or 0.2degree C.

2.1.5 Offset errors

Offset errors are caused by deviation of polarization currents due to increase in temperature or changing in the parameters of the system. These errors are affecting the zero signal input in the way the output will be not equal with zero when the input is maintained constantly to zero volts (e.g. both inputs are shorted). Like gain errors, offset errors are easy corrected using the value stored in the offset register in the time of calibration. For the AD7731 the offset errors after calibration are in the range 0 to 2 microvolts and a drift with temperature of 0.5 microvolts per degree C. Doing a cycled calibration at a short interval of time in the time of operation will eliminate the drift so the only error which will be added to the system is a maximum of 2 microvolts.

2.1.6 Throughput

The throughput is the maximum rate at which the A-D converter can output data values. In general it will be the inverse of the (conversion time + the acquisition time) of the A-D converter. Thus a converter that takes 10 microseconds to acquire and convert will be able to generate about 100 000 samples per second. Throughput may be slowed down, however, by other factors, which prevent data transfer at the full rate. The time to acquire the signal plus the conversion time is related with the programmable parameters of the system and there is a unique value for a set of parameters [10]. Therefore throughput rate will be calculated based on the setting of the digital filter parameters, the output rate and the modulator frequency.
2.1.7 Overall error

The overall error from the point of view of the AD7731 is the sum of offset error with gain error and integral nonlinearity error which is: O.E=2 microvolts+8 microvolts+1.2 microvolts=11.2 microvolts. Or in units of temperature 11.2 microvolts/41 microvolts/degree C=0.27 degree C, almost one quarter of a degree C. These errors are the maximum possible to occur in the system, but because of the special measure, which are considered to be taken in the design I expect for this error to occur at a lower value. To this error will be added an error, which is coming in system from the outside and is generated by the noise. The system is used in a very noisy environment and there are few sources of very high level. The noise will be treated in a special section because of the importance of the problem. If this noise is not rejected properly from the system, the percent of the error will be substantially increased.

2.2 Presentation of AD773

The AD7731 is a complete analog front-end for process control applications. The device has a proprietary programmable gain front end that allows it to accept a range of input signal ranges, including low level signals, directly from a transducer. The sigma-delta architecture of the part consists of an analog modulator and a low pass programmable digital filter, allowing adjustment of filter cutoff, output rate and settling time. See Figure. No 2.2 [9].
The part features three-buffered differential programmable gain analog inputs (which can be configured as five pseudo-differential inputs), as well as a differential reference input [9]. The part operates from a single +5 V supply and accepts seven unipolar analog input ranges: 0 to +20 mV, +40 mV, +80 mV, +160 mV, +320 mV, +640 mV, and +1.28 V, and seven bipolar ranges: ±20 mV, ±40 mV, ±80 mV, ±160 mV, ±320 mV, ±640 mV, and ±1.28 V. The peak-to-peak resolution achievable directly from the part is 16 bits at an 800 Hz output rate. The part can switch between channels with 1 ms settling time and maintain a performance level of 13 bits of peak-to-peak resolution. The serial interface on the part can be configured for three-wire operation and is compatible with microcontrollers and digital signal processors.
Communication between the AD7731 and microcontroller is accomplished using beside of the standard three-wire communication with DI (data in), DO (data out) and SCKL (serial clock) an additional handshaking with the RDY line and RESET line.

The RESET line is used to perform a software reset from the PC, and the RDY is used to synchronize all the reading and writing to part making not necessary to monitor the STATUS register and to increase the time of operation. When a reading of the DATA registers command was send to the ADC, the part will do all the acquisition and all the filtering according with the setting of the FILTER and MODE register. When the data is available to be read, the ADC will assert the RDY pin, and in the same time the RDY bit in the STATUS register will be set.

The microcontroller is monitoring the RDY line and will do the reading of the data according with the status of this. The other option is to read continuous the STATUS register, but this will slow down the process of reading. The AD7731 contains self-calibration and system calibration options and features an offset drift of less than nV/°C and a gain drift less than 2 ppm/°C [9].

2.2.1 Re-Calibration

This A-D converter is able to re-calibrate by itself periodically by measuring a reference voltage, and compensating for offset and gain drift. This is useful for long term monitoring since drift do not accumulate.

For the process of measuring the aluminum alloy temperature, which is a slow varying process, (about half hour) offset and gain errors are expected to occur. If the re-calibration are set to far apart there may appear to be small discontinuities in the recorded data as the re-calibrations occur. (If there is a reading other than zero for a zero condition, then there is an offset error: every reading will be inaccurate by this amount. When the A-D converter is preceded by signal conditioning circuits offset errors need not normally be considered. This design is not using a signal-conditioning scheme therefore the thermocouple cable is connected direct to the AD7731. This means an external calibration scheme will be considerate.
The calibration process involves a very accurate and stable signal for the end of the scale. This is accomplished with a dedicated chip AD780. The AD780 is an ultrahigh precision bandgap reference voltage, which provides a 2.5V output when the part is powered with voltage anywhere between 4 to 36 V see Figure No.2.3 [11]. Low initial error and temperature drift combined with low output noise and the ability to drive any value of capacitance make the AD780 the best choice for the reference voltage for the calibration scheme. The AD780 can be used to source or sink up to 10 mA and can be used in series or shunt mode, thus allowing positive or negative output voltages without external components. This makes it suitable for virtually any high performance reference application. Unlike some competing references, the AD780 has no "region of possible instability."[11].

![Figure No.2.3

2.5 V  High precision Voltage Reference]
The part is stable under all load conditions when a 1μF bypass capacitor is used on the supply. A temperature output pin is provided on the AD780. This provides an output voltage that varies linearly with temperature, allowing the AD780 to be configured as a temperature transducer while providing a stable 2.5 V output. The AD780 is specified for operation from -40°C to +85°C [11].

**Figure No.2.4** shows the block diagram of the system. The thermocouple is connected to the channel 1 of the ADC in differential mode with the 2.5V reference connected to the negative input providing a common mode voltage equal with 2.5V. The internal calibration scheme will provide a mining to read all the errors and to update the offset register with that value. Next when the part will read the thermocouple signal will subtract the contents of the offset register and the result will represent the real value of the temperature.

This scheme for correct errors was enhanced adding an external shunt for calibration. The analog input AI5, AI6 will monitor a precise temperature sensor to measure the temperature of the cold junction. The signal from this input will be stored in microcontroller and used to correct the cold junction offset. The analog digital converter is powered from an independent +5Vdc power supply for the analogical part of the chip and with a different one for the digital part. The power supply for the digital is common for both ADC digital side and microcontroller.

There is a digital ground and an analogical ground in the system. This ground should be connected just in one point to avoid forming loops and consequently undesired voltage in the system. The ADC have own clock signal at a speed of 4.9152 MHz from the on board oscillator. The signal from the thermocouple is acquired to the AI1, AI2 with a selectable rate. For reasons, which will be shown in the design, the rate of acquiring the signal is the 400 Hz or at an interval of 2.5 milliseconds. The input is connected to a buffer followed by a programmable gain amplifier. The buffer will increase the input impedance to 300 Kohm, which allows to interface sensors, which
exhibit increase output resistance. The programmable gain amplifier allows scaling the signal with a scale factor tacked from the GAIN register and updated after every part calibration. Consequently, this will correct the gain errors. The same effect will have the offset data over the amplifier, lifting or lowering the floor of the signal, therefore correcting the offset errors detected after calibration.

Figure No. 2.4  
Block Diagram of the System
The programmable gain amplifier is followed by the Sigma–Delta modulator. The modulator has a programmable resolution between 24 bits and 15 bits. The resolution is selected based on the noise requirement and the output update rate. For the reasons imposed by the performance of the system this resolution has selected to 16 bits therefore allowing a transfer rate of 400Hz at a noise figure of 335 nV.

The filter bank, which follows the modulator, could be skipped if the MODE register is programmed accordingly. As well the chopper feature, which is implemented in this stage, could be disabled or not. If not disabled it will enhance the performance of the part, removing the drift of the signal in the time of operation. If the filter is enabled, it allows rejection of undesired noise from the signal and also improves the overall system performance removing the noise produced by quantization. Further will be presented in detail the filters in the design section.

After the signal is acquired and processed in the amplifier and filter bank, the resulting data is stored in the DATA register. This register could be programmed to store 2 bytes or 3 bytes of data. From this register the data is moved to microcontroller via SPI (serial peripheral interface), which is a high-speed synchronous serial communication interface controlled by the microcontroller clock. The SPI interface is programmable to communicate with a speed from 31.3KHz to 4MHz in 8 steps. For the purpose of programming and testing the part the speed was set to 500KHz. For the time the part will work for acquiring signal from thermocouple, the speed will be increase to a better value. Data could be transferred from the ADC to microcontroller in a continuous mode or in just one time reading. This is programmable also from the MODE register.

From the ADC, the transfer of information is performed via a three state buffers inserted between microcontroller and ADC, to ensure that the noise from the switched bus of the microcontroller will not flow into the AD7731 and to depreciate the quality of conversion.

In microcontroller data is stored in the RAM area of memory and further is correlated with the data processed from the reading of the temperature of the cold junction.

Software compensation is performed using a table of vectors with prestored value of the Sebeck coefficients and multiplied with the updated temperature of the junction.
The microcontroller also performs a cyclic calibration of the part to avoid deterioration of the parameters of the system against time and temperature. Processed data from the microcontroller is moved serially to the PC using the serial communication interface (SCI). The transfer of this data is performed at 9600 bauds even if there is possibility to increase the speed to 38400 bauds. After testing the program this speed will be increased to a higher rate if possible. This will be mentioned in the Tests and Conclusions Chapter.

Other features of the microcontroller should be mentioned here: The clock of this part is internally at 8 MHz and is software adjustable to increase the accuracy of the timed operations. 68HC12 is a 16 bits microcontroller, this means the address spaces is 2 squared by16 allowing an increase zone of RAM (2Kbits) a considerable amount of electrical erasable flash Eeprom (32767 Kbits), as well a zone of 767 bits Eeprom.

The part feature beside of the serial communication interface SCI and the peripheral communication interface SPI a complex internal independent timer system and an enhanced counter system, an analog digital converter port with 8 channels, four PWM pulse with modulation outputs, seven digital I/O ports at 8 pins every configurable like ordinary I/O or like special function. This part also has a special port for interfacing in a multichip network BDLC (byte data link communication).

The set of instructions of this microcontroller exhibit enhanced feature, allowing fuzzy logic and high-level mathematic operations.

The program for this part is done in assembly language using the TeXaS integrated development environment, which allows writing the program in assembly language, also to simulate and debug application offline. 68HC12 allows the program and also debugging the part live via BDM (background debug mode), which is a new feature for this type of microcontroller. This is a one pin communication, with the part and is done in the time of running of the microcontroller when the CPU is hold in dead times or when there is a reading from the memory. In those moments the BDM is reading the registers and the state of the program, which is running live. This is a very powerful tool, which easy replaces the traditional logic analyzer used for these operations.

The program for this microcontroller is attached in Appendix E, at the end of the project.
Data received from ADC via microcontroller is stored in the PC under the control of a monitor routine written in Visual C++. This program is performing the handshaking between the PC and microcontroller, allowing configuring all the parameters of the analog-digital-converter, the digital filters, to perform calibration and to acquire data for one repeated reading or with a continuous reading. The data is stored in an array of binary file from where is possible to be exported to Matlab environment and to perform all the necessary processing, which Matlab and Simulink allow to be performed.
CHAPTER 3

3.1 Design of the Calibration System

The analog digital converter AD7731 is equipped with a two system of calibration. First Internal Calibration for the part itself that use internal calibrated signals for all the combination of channels and levels available. The second System Calibration is for the system and this will imply external signals for the zero level and for the full scale for the selected channel (in our case the thermocouple and the cold junction channel). The cold junction is implemented on the channel 3 of the ADC.

For the compensation of the cold junction, the software technique was preferred, which use the temperature picked up from the point where is implemented the cold junction (thermocouple connection to the system), and further process this signal to find the level of the supplementary voltage which the cold junction will add and make the necessary corrections.

The sensor used to do the reading of the cold junction temperature is an Analog Devices AD592 high precision transducer that output a current proportional with the absolute temperature. The characteristic of this sensor is very linear with temperature and the errors, which will output over the 0 to 105 degree Celsius, are less than 0.1%, refer to Figure 3.1. [12]. For a wide range of supply voltages the transducer acts as a high impedance temperature dependent current source of 1 μA/K. The AD592 can be employed in applications between -25°C and +105°C where conventional temperature sensors (i.e., thermistor, RTD, thermocouple, diode) are currently being used [12]. Typical application areas include: automotive temperature measurement and control, system monitoring, industrial temperature control, thermocouple cold junction compensation, board-level electronics temperature diagnostics, temperature readout options in instrumentation, and temperature correction circuitry for precision electronics.
AD592 is immune to voltage drops and voltage noise over long lines due to its high impedance current output. The theory of operation of the entire calibration scheme is based on the operation of this sensor and the complex structure of the calibration system of the ADC. The output current of the transducer versus absolute temperature is linear over the entire range. The Figure No. 3.2 [12] show this from the −45 to +125 degree Celsius. The slope of this is 1 microamper/1K. This means that for zero degrees Celsius the part will output 273 microamperes.

To obtain the output current versus degree Celsius should be subtracted from the resultant current the value for zero degree, which is: (I -273) microamperes. The transducer is injecting this current into a calibrated resistor. For this application the resistor R_{c} is choose to have 3665 Ohm. For a 25 degree Celsius this will inject a current of 273+25 microamperes or the voltage drop on this resistor will be 3665x298x10^{-6}=1.092V.
Figure No. 3.2
Output Current Versus Temperature

For the maximum ambient temperature (about 50 degree Celsius) the voltage drop across the resistor will be $U_{\text{max}}=3665\times(273+50)\times10^{-6}=1.18\text{V}$. This will make suitable to be used the 1.28V range for the channel No.3 of the ADC.

To find the temperature of the sensor the following formula is applicable:
$T=[U- (273\times10^{-6}\times3665)]/3665\times10^{-6}$. Ex. If the voltage read over the resistor is 1.1V then applying the formula will obtain: $(1.1-1.000545)/0.003665=27.13$ degree Celsius.

One feature of the microcontroller is to process the data and to be able to perform mathematical operations. This entire algorithm could be implemented in the
microcontroller program in the way that the cold junction voltage will be calculate and scaled to obtain the temperature in degree Celsius then the thermocouple signal will be corrected subtracting the temperature of the cold junction multiplied with the Sebeck coefficient.

This choice will impose that the reading of the ADC to be carried out with two data register to be compatible with the 16 bits capability of the 68HC12 microcontroller. The second choice is to perform all the calculation in the PC where the mathematic operations could be effectuated in floating point and the precision to be increased. Also the reading of the ADC could be carried out with three data register, therefore the 24 bits resolution will increase the overall precision. The second choice was preferred for the seek of accuracy.

To achieve better accuracy in acquiring the transducer signal between the voltages picked up over the resistor and the ADC channel it was inserted a follower amplifier AD711.

This amplifier is a high speed, BiFET op amplifier with a very low offset and drift and a very high slew rate of 16V/microseconds [13]. The amplifier will increase the input impedance to the order of $10^7$ Mohm, making the reading very accurate. To achieve high accuracy in the reading and processing of the signal a part calibration and a zero system calibration is performed before the reading of the temperature will be done. The temperature sensor is mounted between the two connectors of the thermocouple cable, to ensure that the junction temperature is the same than the one picked up from the sensor.

To increase the heat transfer a silicone past was immersed between the transducer and the PCB. Appendix G shows the complete calibration diagram. To do the zero scale calibration for the system the sensor pins should be shorted to ensure the zero volts at the input of the channel.

The part is designed with a system for internal calibration and one for external calibration. The internal calibration is performed for the zero internal signals with an internally generated shunt. For the full scale calibration the part will generate a precise internal voltage for the full scale of the selected range. This voltage is scaled from the reference voltage (Vref=2.5V). First internal calibration is performed for the zero scale and this will take into account all the internal offset and drift based on the temperature of
the part and the setting for the filter and mode registers (settling time, gain, unipolar/bipolar operation).

The AD7731 provides a number of calibration options that can be programmed via the MD2, MD1 and MD0 bits of the Mode Register. The operation of the internal zero scale calibration will be performed by the part based on the command embedded in the mode register for the specific type of calibration and specific channel. Ex. For the channel 3 for the 1.28V unipolar operation the mode register will be updated with MODE1: 10010000 or 90H, and MODE2:01110110 or 76H. The calibration will be performed with the external channel shorted to ensure better precision for the zero volts level. Calibration cycle may be initiated at any time by writing to the bits of the Mode Register. The configuration for these bits are presented in the Table 3.1 [9] and the setting of these bits will define the type of conversion.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MD2</td>
<td>MD1</td>
<td>MD0</td>
<td>B/U</td>
<td>DEN</td>
<td>D1</td>
<td>D0</td>
<td>WL</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MR7</td>
<td>MR6</td>
<td>MR5</td>
<td>MR4</td>
<td>MR3</td>
<td>MR2</td>
<td>MR1</td>
<td>MR0</td>
</tr>
<tr>
<td>HIREF</td>
<td>RN2</td>
<td>RN1</td>
<td>RN0</td>
<td>B0</td>
<td>CH2</td>
<td>CH1</td>
<td>CH0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table No.3.1

Mode Register Configuration

The way the calibration are selected with the MD2, MD1, and MD0 bits in the Mode register and are outlined in the Table No. 3.2 [9]. The available settings for these bits and the effect over the behavior of the part will be discussed next.
<table>
<thead>
<tr>
<th>MD2</th>
<th>MD1</th>
<th>MD0</th>
<th>MODE OF OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SYNK (IDLE) MODE</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>CONTINUOUS CONVERSION MODE</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>SINGLE CONVERSION MODE</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>POWER DOWN (STANDBY) MODE</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INTERNAL-ZERO-SCALE-CALIBRATION</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>INTERNAL-FULL-SCALE-CALIBRATION</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>SYSTEM-ZERO-SCALE-CALIBRATION</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>SYSTEM-FULL-SCALE-CALIBRATION</td>
</tr>
</tbody>
</table>

**Table No.3.2**

Operating Mode for the AD7731

**MR12** B/U  Bipolar/Unipolar Bit. A 0 in this bit selects bipolar operation and the output coding is 00...000 for negative full-scale input, 10...000 for zero input and 11...111 for positive full-scale input. A 1 in this bit selects unipolar operation and the output coding is 00...000 for zero input and 11...111 for positive full-scale input.

**MR11** DEN  Digital Output Enable Bit. With this bit at 1, the AIN3/D1 and AIN4/D0 pins assume their digital output functions and the output drivers connected to these pins are enabled. In this mode, the user effectively has two port bits that can be programmed over the serial interface.

**MR10–MR9** D1–D0  Digital Output Bits. These bits determine the digital outputs on the AIN3/D1 and AIN4/D0 pins respectively when the DEN bit is a 1. For example, a 1 written to the D1 bit of the Mode Register (with the DEN bit also a 1) will
put logic 1 on the AIN3/D1 pin. This logic 1 will remain on this pin until a 0 is written to the D1 bit (in which case, the AIN3/D1 pin goes to a logic 0) or writing a 0 to the DEN bit disables the digital output function.

**MR8** WL Data Word Length Bit. This bit determines the word length of the Data Register. A 0 in this bit selects 16-bit word length when reading from the data register (i.e., RDY returns high after 16 serial clock cycles in the read operation). A 1 in this bit selects 24-bit word length for the Data Register.

**MR7** HIREF High Reference Bit. This bit should be set in accordance with the reference voltage, which is being used on the part. If the reference voltage is 2.5 V, the HIREF bit should be set to 0. If the reference voltage is 5 V, the HIREF bit should be set to a 1. With the HIREF bit set correctly for the appropriate applied reference voltage, the input ranges are 0 mV to +20 mV, +40 mV, +80 mV, +160 mV, +320 mV, +640 mV and +1.28 V for unipolar operation and ±20 mV, ±40 mV, ±80 mV, ±160 mV, ±320 mV, ±640 mV and ±1.28 V for bipolar operation. It is possible for a user with a 2.5 V reference to set the HIREF bit to a 1. In this case, the part is operating with a 2.5 V reference but assumes it has a 5 V reference. As a result, the input ranges on the part become 0 mV to +10 mV through 0 mV to +640 mV for unipolar operation and ±10 mV through ±640 mV for bipolar operation. However, the output noise from the part (in nV) will remain unchanged so the resolution of the part (in LSBs) will reduce by 1.

**MR6–MR4** RN2–RN0 Input Range Bits. These bits determine the analog input range for the selected analog input. The different input ranges are outlined in Table XII. The table is valid for a reference voltage of 2.5 V with the HIREF bit at 0 or for a reference voltage of 5 V with the HIREF bit at logic 1. The selection for these bits is outlined in Table No.3.3 [9]
<table>
<thead>
<tr>
<th>RN2</th>
<th>RN1</th>
<th>RN0</th>
<th>B/U Bit=0</th>
<th>B/U Bit=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-20 mV to +20 mV</td>
<td>0 mV to 20 mV</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-20 mV to +20 mV</td>
<td>0 mV to 20 mV</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-40 mV to +40 mV</td>
<td>0 mV to 40 mV</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-80 mV to +80 mV</td>
<td>0 mV to 80 mV</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-160 mV to +160 mV</td>
<td>0 mV to 160 mV</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-320 mV to +320 mV</td>
<td>0 mV to 320 mV</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-640 mV to +640 mV</td>
<td>0 mV to 640 mV</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1.28 V to +1.28 V</td>
<td>0 mV to 1.28 V</td>
</tr>
</tbody>
</table>

**Table No.3.3**

**Input Range Selection of the AD7731**

**MR3** BO Burnout Current Bit. A 1 in this bit activates the burnout currents. When active, the burnout currents connect to the selected analog input pair, one source current to the AIN (+) input and one sink current to the AIN (−) input. A 0 in this bit turns off the on-chip burnout currents.

**MR2–MR0** CH2–CH0 Channel Select. These three bits select a channel either for conversion or for access to calibration coefficients as outlined in Table XIII. There are three pairs of calibration registers on the part. In fully differential mode, the part has three input channels so each channel has its own pair of calibration registers. In pseudodifferential mode, the AD7731 has five input channels with some of the input channel combinations sharing calibration registers. With CH2, CH1 and CH0 at logic 1, the part looks at the AIN6 input internally shorted to itself. The selection of the Channels is outlined in Table No.3.4 [9]
<table>
<thead>
<tr>
<th>CH2</th>
<th>CH1</th>
<th>CH1</th>
<th>AIN (+)</th>
<th>AIN (-)</th>
<th>TYPE</th>
<th>CALIBRATION REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>AIN1</td>
<td>AIN6</td>
<td>PSEUDO-DIFF.</td>
<td>REGISTER PAIR 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>AIN2</td>
<td>AIN6</td>
<td>PSEUDO-DIFF.</td>
<td>REGISTER PAIR 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>AIN3</td>
<td>AIN6</td>
<td>PSEUDO-DIFF.</td>
<td>REGISTER PAIR 2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>AIN4</td>
<td>AIN6</td>
<td>PSEUDO-DIFF.</td>
<td>REGISTER PAIR 2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>AIN5</td>
<td>AIN2</td>
<td>FULLY-DIFF.</td>
<td>REGISTER PAIR 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>AIN6</td>
<td>AIN4</td>
<td>FULLY-DIFF.</td>
<td>REGISTER PAIR 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>AIN7</td>
<td>AIN6</td>
<td>FULLY-DIFF.</td>
<td>REGISTER PAIR 2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>AIN6</td>
<td>AIN6</td>
<td>TEST-MODE</td>
<td>REGISTER PAIR 2</td>
</tr>
</tbody>
</table>

Table No.3.4
Channel Selection of AD7731

Calibration on the AD7731 removes offset and gains errors from the device. The AD7731 gives the user access to the on-chip calibration registers allowing the microprocessor to read the device’s calibration coefficients (Offset Ch1, Ch3 Register 1,2,3 and Gain Ch1, Ch3 with another 3 Register), and also to write its own calibration coefficients to the part from prestored values in the monitor program if the condition for the acquisition will be changed. This gives to the monitor program much greater control over the AD7731’s calibration procedure. It also means that by comparing the coefficients after calibration with prestored values in program, the user can verify that the device has correctly performed its calibration.

The values in these calibration registers are 24 bits wide. In addition, it is possible to adjust the span and offset for the part testing for a variety of ambient temperature and work condition. This can be used as a test method to evaluate the noise performance of the part with no external noise sources. In this mode, the AIN6 input should be connected to an external voltage within the allowable common-mode range for the part. The power-on/default status of these bits is 1, 0, and 0. Internally in the AD7731, the coefficients are normalized before being used to scale the words coming out of the digital filter.
The offset calibration register contains a value which, when normalized, is subtracted from all conversion results.

The gain calibration register contains a value which, when normalized, is multiplied by all conversion results. The offset calibration coefficients are subtracted from the result prior to the multiplication by the gain coefficient.

The AD7731 offers self-calibration or system calibration facilities. For full calibration to occur on the selected channel, the on-chip microcontroller must record the modulator output for two different input conditions. These are “zero-scale” and “full-scale” points. Performing a conversion on the different input voltages provided to the input of the modulator during calibration derives these points. The result of the “zero-scale” calibration conversion is stored in the Offset Calibration Registers for the appropriate channel.

The result of the “full-scale” calibration conversion is stored in the Gain Calibration Register for the appropriate channel. With these readings, the microcontroller can calculate the offset and the gain slope for the input-to-output transfer function of the converter.

Internally, the part works with 33 bits of resolution to determine its conversion result of either 16 bits or 24 bits. The sequence in which the zero-scale and full-scale calibration occurs depends upon the type of full-scale calibration being performed. The internal full-scale calibration is a two-step calibration that alters the value of the Offset Calibration Register. Thus, there **must be performing** a zero-scale calibration (either internal or system) after an internal full-scale calibration to correct the Offset Calibration Register contents. When using system full-scale calibration, it is recommended that the zero-scale calibration (either internal or system) to be performed first. The setting of the Filter Register will also have direct effect over the calibration. SKIP, CHP, and FAST are bits of this register and the meaning of these bits will be further discussed in the Filtering section in Chapter No.4

Calibration time is the same regardless of whether the SKIP mode is enabled or not. This is because the SKIP bit is ignored and the second stage filter is included in the calibration cycle. This is done to derive more accurate calibration coefficients. If the subsequent operating mode is with CHP = 0, the calibration should be performed with
CHP = 0 so the offset calibration coefficient and the subsequent conversion offsets are consistent.

Since the calibration coefficients are derived by performing a conversion on the input voltage provided, the accuracy of the calibration can only be as good as the noise level which the part provides in normal mode. To optimize the calibration accuracy, it is recommended to calibrate the part at its lowest output rate where the noise level is lowest. The coefficients generated at any output update rate will be valid for all selected output update rates. This scheme of calibrating at the lowest output update rate does mean that the duration of calibration is longer.

3.1.1 System Full-Scale Calibration

A system full-scale calibration is initiated on the AD7731 by writing the appropriate values (1, 1, 1) to the MD2, MD1 and MD0 bits of the Mode Register see Table No.3.2 [9]. System full-scale calibration is performed using the system's positive full-scale voltage. This voltage (+1.228V AD780) full-scale voltage must be set up before the calibration is initiated, and it must remain stable throughout the calibration step. The system full-scale calibration is performed at the selected gain as per the RN2, RN1, RN0 bits in the Mode Register see Table No.3.3 [9].

The duration time of the calibration depends upon the CHP bit of the Filter Register. With CHP = 1, the duration is 22 × 1/Output Rate; with CHP = 0, the duration is 24 × 1/Output Rate. For the Output Rate see the Filtering section in the next chapters.

At this time the MD2, MD1 and MD0 bits in the Mode Register return to 0, 0, 0 (Sync or Idle Mode for the AD7731). The RDY line goes high when calibration is initiated and returns low when calibration is complete. Therefore this line is monitored in the program to sense the end of the calibration. An additional delay is added to ensure a better reading of the gain due to the longer time needed for the output to settle. Note that the part has not performed a conversion at this time; it has simply performed a full-scale calibration and updated the Gain Calibration Register for the selected channel. Then must write 0, 0, 1 or 0, 1, 0 to the MD2, MD1, and MD0 bits of the Mode Register to initiate a single or a continuous conversion. If RDY is low before (or goes low during) the calibration
command will write to the Mode Register, it may take up to one modulator cycle (MCLK IN/16) before RDY goes high to indicate that calibration is in progress. Therefore, RDY should be ignored for up to one modulator cycle after the last bit of the calibration command is written to the Mode Register. A delay was also added here to ensure that the reading of the RDY line would not find this line asserted yet.

The system full-scale calibration needs to be performed as one part of a two part full calibration. When performing a two-step full calibration, care should be taken as to the sequence in which the two steps are performed. A system full-scale calibration should not be carried out unless the part contains valid zero-scale coefficients (offset updated). Therefore, an internal zero-scale calibration or a system zero scale calibration must be performed before the system full-scale calibration when a full two-step calibration operation is being performed.

3.1.2 Internal Full-Scale Calibration

An internal full-scale calibration is initiated on the AD7731 by writing the appropriate values (1, 0, 1) to the MD2; MD1 and MD0 bits of the Mode Register see Table No.3.2 [9]. In this calibration mode, the full-scale point used in determining the calibration coefficients is with an internally generated full-scale voltage. This full-scale voltage is derived from the reference voltage for the AD7731 and the PGA is set for the selected gain (as per the RN2, RN1, RN0 bits in the Mode Register, see Table No.3.3 [9] for this internal full-scale calibration conversion.

Normally, the internal full-scale calibration is performed at the required operating output range. When operating with a 20 mV or 40 mV input range, it is recommended that internal full-scale calibrations are performed on the 80 mV input range.

The internal full-scale calibration is a two-step sequence, which runs when an internal full-scale calibration command is written to the AD7731.

One part of the calibration is a zero-scale calibration and as a result, the contents of the Offset Calibration Register are altered during this Internal Full-Scale Calibration. There must be performed, a zero-scale calibration (either internal or system) AFTER the internal full-scale calibration. This means that internal full-scale calibrations cannot be
performed in isolation. Again duration time of the calibration depends upon the CHP bit of the Filter Register. With CHP = 1, the duration is 44 × 1/ Output Rate; with CHP = 0, the duration is 48 × 1/Output Rate. At this time the MD2, MD1 and MD0 bits in the Mode Register return to 0, 0, 0 (Sync or Idle Mode for the AD7731).

The RDY line goes high when calibration is initiated and returns low when calibration is complete. Note that the part has not performed a conversion at this time. The user must write 0, 0, 1 or 0, 1, 0 to the MD2, MD1, and MD0 bits of the Mode Register to initiate a conversion. If RDY is low before (or goes low during), the calibration commands write to the Mode Register. It may take up to one modulator cycle (MCLK IN/16) before RDY goes high to indicate that calibration is in progress. Therefore, RDY should be ignored for up to one modulator Cycle after the last bit of the calibration command is written to the Mode Register.

3.1.3 System Zero-Scale Calibration

System calibration allows the AD7731 to compensate for system gain and offset errors as well as its own internal errors. System calibration performs the same slope factor calculations as self-calibration but uses voltage values presented by the system to the AIN inputs for the zero points. A system zero-scale calibration is initiated on the AD7731 by writing the appropriate values (1, 1, 0) to the MD2, MD1 and MD0 bits of the Mode Register.

In this calibration mode with a unipolar input range, the zero-scale point used in determining the calibration coefficients is the bottom end of the transfer function. The system’s zero-scale point is applied to the AD7731’s AIN input before the calibration step and this voltage must remain stable for the duration of the system zero-scale calibration. It was used the short-circuit between the two inputs to establish zero level reference, using the Shunt Relay No2 for Channel 1 and Shunt Relay No.3 for Channel 3. The PGA is set for the selected gain (as per the RN2, RN1, RN0 bits in the Mode Register) for this system zero-scale calibration conversion. The allowable range for the system zero-scale voltage is discussed in the Span and Offsets Section.
The duration time of the calibration depends upon the CHP bit of the Filter Register. With CHP = 1, the duration is $22 \times 1/\text{Output Rate}$; with CHP = 0, the duration is $24 \times 1/\text{Output Rate}$. At this time the MD2, MD1 and MD0 bits in the Mode Register return to 0, 0, 0 (Sync or Idle Mode for the AD7731).

The RDY line goes high when calibration is initiated and re-turns low when calibration is complete. Note, the part has not performed a conversion at this time; it has simply performed a zero-scale calibration and updated the Offset Calibration Register for the selected channel. There should be write 0, 0, 1 or 0, 1, 0 to the MD2, MD1, MD0 bits of the Mode Register to initiate a conversion. If RDY is low before (or goes low during) the calibration command writes to the Mode Register, it may take up to one modulator cycle (MCLK IN/16) before RDY goes high to indicate that calibration is in progress. Therefore, RDY should be ignored for up to one modulator cycle after the last bit of the calibration command is written to the Mode Register.

For bipolar input ranges in the system zero-scale calibrating mode, the sequence is very similar to that just outlined. In this case, the zero-scale point is the mid-point of the AD7731’s transfer function.

The system zero-scale calibration needs to be performed as one part of a two part full calibration. When performing a two-step full calibration, care should be taken as to the sequence in which the two steps are performed. If the system zero-scale calibration is one part of a full system calibration, it should take place before a system full-scale calibration. If it takes place in association with an internal full-scale calibration, this system zero-scale calibration should be performed after the full-scale calibration.

3.1.4 Internal Zero-Scale Calibration

An internal zero-scale calibration is initiated on the AD7731 by writing the appropriate values (1, 0, 0) to the MD2, MD1 and MD0 bits of the Mode Register. In this calibration mode with a unipolar input range, the zero-scale point used in determining the calibration coefficients is with the inputs of the differential pair internally shorted. The PGA is set for the selected gain (as per the RN2, RN1, RN0 bits in the Mode Register) for this internal zero-scale calibration conversion.
The duration time of the calibration depends upon the CHP bit of the Filter Register. With CHP = 1, the duration is $22 \times 1/\text{Output Rate}$; with CHP = 0, the duration is $24 \times 1/\text{Output Rate}$.

At this time the MD2, MD1 and MD0 bits in the Mode Register return to 0, 0, 0 (Sync or Idle Mode for the AD7731). The RDY line goes high when calibration is initiated and re-turns low when calibration is complete. Note that the part has not performed a conversion at this time; it has simply performed a zero-scale calibration and updated the Offset Calibration Register for the selected channel. There should be written 0, 0, 1 or 0,1,0 to the MD2, MD1, and MD0 bits of the Mode Register to initiate a conversion. If RDY is low before (or goes low during) the calibration commands write to the Mode Register, it may take up to one modulator cycle (MCLK IN/16) before RDY goes high to indicate that calibration is in progress.

Therefore, RDY should be ignored for up to one modulator cycle after the last bit of the calibration command is written to the Mode Register.

For bipolar input ranges in the internal zero-scale calibrating mode, the sequence is very similar to that just outlined. In this case, the zero-scale point is exactly the same as above but since the part is configured for bipolar operation, the output code for zero differential input is 800000 Hex in 24-bit mode. The internal zero-scale calibration needs to be performed as one part of a two-step full calibration. However, once a full calibration has been performed, additional internal zero-scale calibrations can be performed by themselves to adjust the part's zero-scale point only.

When performing a two-step full calibration, care should be taken as to the sequence in which the two steps are performed. If the internal zero-scale calibration is one part of a full self-calibration, then it should take place after an internal full-scale calibration. If it takes place in association with a system full-scale calibration, then this internal zero-scale calibration should be performed first. The Calibration electrical diagram is presented in the Appendix G.
CHAPTER 4

4.1 Signal versus Noise

The thermocouple signal is an analog signal. This signal is generated at very low levels and the Ratio Signal – Noise is small. Therefore the signal is suffering a corruption from the other external signals like EMF (electromagnetic interference), spikes and switching noise, especially drivers and electrical motors. These harmful signals are rejected mostly with proper shielding of the incoming cable signal. Also screens and special hardware measure adopted in the PCB design will improve the rejection of the noise. Some times low pass filters are inserted in front of the ADC in order to reject this signals especially the noise, which is residing in the higher range of the spectrum.

The 60 Hz signal from the power supply is still present and is very strong because of the EMF from the electrical lines and this signal should be rejected with the system of filters available on the part. On the other side new undesirable signal appear on the process of conversion of the signal from analog to digital. The process of conversion is by his nature noisy and one type of noise is the quantization noise. In every second the ADC is taking a number of samples and this is called Output Rate. This number is programmable and a set of tables available from the Analog Devices outline the different setting available for the Output Rate, related with the noise output from part, and the rest of the conversion parameters. For a specific Output Rate, a noise will be observed on that frequency and on the second and third harmonics of that frequency.

For example if the part will deliver data whit an Output Rate of 200 Hz than the noise will be present at this frequency and also at 400Hz and 600Hz. The amplitude of the noise is diminishing with the increase of the harmonics number. Another type of noise, which is coming from the process of conversion, is the noise produced by Aliasing. This type of noise is generated from the difference between the sampling frequency (which is at least 2 time than of the highest frequency in the band of interest) and the other frequency, which could be present (i.e. spikes, EMF) and lies between the band of interest and the sampling frequency.
Figure No.4.1 [15]. shows a signal with a very low pure sinusoidal frequency which spectrum is composed with its fundamental component. Because of not rejecting the signal harmonics which lies between the signal and the sampling frequency aliasing will occur and change the form of original signal. Aliasing will bring unwanted signal in all the band of frequency, down to the DC for the signal of interest. One method to avoid aliasing is to sample with a frequency much higher than the double of the highest frequency in the band of interest. This technique is call oversampling. By his nature the Sigma–Delta analog digital converter is oversampling and this will make not necessary a low-pass filter inserted before the ADC.
Figure No. 4.1

Effect of aliasing of a sinusoidal signal
With its own non-filtrate harmonics.
4.2 Sigma-Delta Modulator

A sigma-delta ADC generally consists of two main blocks, an analog modulator and a digital filter. In the case of the AD7731, the analog modulator consists of a difference amplifier, an integrator block, a comparator and a feedback loop with a DAC as illustrated in Figure No.4.2 [9]. In operation, the analog signal sample is fed to the difference amplifier along with the output of the feedback DAC. The difference between these two signals is integrated and fed to the comparator. The output of the comparator provides the input to the feedback DAC so the system functions as a negative feedback loop that tries to minimize the difference signal.

The digital data that represents the analog input voltage is contained in the duty cycle of the pulse train appearing at the output of the comparator. This duty cycle data can be recovered as a data word using the digital filter. The sampling frequency of the modulator loop is many times higher than the bandwidth of the sampling frequency Fs, where Fs is the Nyquiste frequency for the band of interest. This will bring the quantization noise out of the low frequency band where the signal of interest resides. The integrator in the modulator shapes the quantization noise (which results from the analog to digital conversion) so that the noise is pushed towards one half of the modulator frequency. The digital filter (FIR) then band limits the response to a frequency significantly lower than one half of the modulator frequency.

![Figure No.4.2](image)

**Figure No.4.2**

Block Diagram of the Sigma-Delta Modulator
The analysis of the Delta–Sigma modulator is best understood in frequency domain. The integrator behave like an analalogical filter with the output amplitude proportional with \(1/f\) therefore the characteristic of this is a high pass filter moving the quantization noise far from the low frequency zone where the signal of interest is located.

For a gain of 1 the output \(y=(x-y)/f + Q\) where \(Q\) = quantization noise. Or after rearranging \(y=x/(f+1)+Qf/(f+1)\). Doing the limit for this function when \(\text{'}f\text{'}\) approach zero the output approach \(x\) and \(Q\) will vanish. The signal will approach to DC the noise will be shaped and pushed on the high frequency domain. After this process a filtering will remove complete this noise. Therefore two banks of filters follow the Sigma-Delta modulator.

### 4.3 Digital Filtering

#### 4.3.1 Filter Architecture. The output of the modulator feeds directly into the digital filter. This digital filter consists of two portions, a first stage filter and a second stage filter. The cutoff frequency and output rate of the filter are programmable.
4.3.2 The First Stage Filter

The first stage filter is a low-pass, sinc 3 or $(\sin x/x)$ 3 filter whose primary function is to remove the quantization noise introduced at the modulator. Figure No.4.4 [15] illustrates and sinc function the sinc function is defined as:

\[
\text{Sinc}(a) = \frac{\sin(\pi a)}{\pi a}
\]

The general form is $\sin(x)/x$. In other words, the sinc is a sine wave that decays in amplitude as $1/x$. Sinc filters are used to separate one band of frequencies from another. They are very stable, and can be pushed to incredible performance levels. These exceptional frequency domain characteristics are obtained at the expense of poor performance in the time domain, including excessive ripple and overshoot in the step response.

The general form for the positive side of the graph is illustrated in Figure No.4.5. [15] The particularity of this function is that the Fourier Transform in time domain of this function is a square. Therefore this is the reason this function is used frequently in special in the Sigma–Delta Analog digital converters. The characteristic of the low–pass filter that is using sinc function approach an ideal characteristic.

![Sinc function representation](image)

Figure No.4.4

Sinc function representation
The notches between the mains leaves are important from one point of view. On the part this notches are programmable and could be adjusted on the points where possible undesirable frequencies reside If this function is squared or cubed the characteristic obtained allow a much better attenuation on the notches points. A compare of different power of this function is outlined on the Figure No. 4.5 [15]. The bigger the power of the filter, the bigger attenuation is obtained on the notch. Actually for a power of 4 the attenuation is greater than 140 dB, and for a power of 3 this is still around 120 dB. The filter exhibit a pass band composed of several leaves bordered by narrows notches. If for example the 50 Hz power frequency is a major factor of noise than the first notches could be tuned on this frequency and the rest of this notches will comes at 100Hz, 150Hz and so on rejecting also the harmonics for this frequency. This is the reason that this filter is used in the ADC technique.

Figure No. 4.5

Comparison between powers of SINC
4.3.3 The second stage filter

The second stage filter has three distinct modes of operation. The first option is where it is bypassed completely such that the only filtering provided on the AD7731 is performed by the first stage sinc 3 filters. The second is where it provides a low-pass 22-tap FIR filter, which processes the output of the first stage filter. The third option is to enable FAST Step mode. In this mode, when a step change is detected on the analog input or the analog input channel switched, the second stage filter enters a mode where it performs a variable number of averages for some time after the step change and then the second stage filter switches back to the FIR filter. The AD7731 has two primary modes of operation, chop mode (CHP = 1) and nonchop mode (CHP = 0). The AD7731 alternatively reverses its inputs with CHP = 1, and alternate outputs from the first stage filter have a positive offset and negative offset term included. With CHP = 0, the input is never reversed and the output of the first stage filter includes an offset which is always of the same polarity. The operation mode can be changed to achieve optimum performance in various applications. The CHP bit should generally be set to 0 when using the AD7731 in applications where higher throughput rates are a concern or in applications where the reduced rejection at the chopping frequency in chop mode is an issue.

The part should be operated with CHP = 1 when drift, noise rejection and optimum EMI rejection are important criteria in the application. The output update rate of the AD7731 is programmed using the SF bits of the Filter Register. With CHP = 0, the output update is determined by the relationship:

Output Rate = \( f_{\text{MOD}} \times 1 \ SF \quad (\text{CHP} = 0) \)

where SF is the decimal equivalent of the data loaded to the SF bits of the Filter Register and fMOD is the modulator frequency and is 1/16th of the master clock frequency [9]. With CHP = 1, the output update is determined by the relation-ship:

Output Rate = \( f_{\text{MOD}} \times 1 \times 3 \times SF \quad (\text{CHP} = 1) \)

where SF is the decimal equivalent of the data loaded to the SF bits of the Filter Register and fMOD is the modulator frequency and is 1/16th of the master clock frequency [9]. Thus for a given SF word the output rate from the AD7731 is three times faster with CHP = 0 than CHP = 1. The various filter stages and options are discussed in the following sections.
4.3.4 First Stage Filter/SKIP Mode Enabled (SKIP = 1)

With SKIP mode enabled, the only filtering on the part is the first stage filter. The frequency response for this first stage filter is shown in Figure No.4.6 [9]. The response of this first stage filter is similar to that of an averaging filter but with a sharper roll-off. With CHP = 0, the output rate for the filter corresponds with the positioning of the first notch of the filter's frequency response. Thus, for the plot of Figure No.4.6 [9] where the output rate is 600 Hz (FCLK IN = 4.9152 MHz and SF = 512), the first notch of the filter is at 600 Hz.
4.3.5 Nonchop Mode (SKIP = 1, CHP = 0)

With CHP = 0, the input chopping on the AD7731 is disabled and any offset content in the samples to the first stage filter are all of the same polarity. When using the part in SKIP mode, the user can take the output from the AD7731 directly. Time to the first output for the part is $3 \times 1$/Output Rate in this mode. Table 4.1 [9] summarizes the settling time and subsequent throughput rate for the various different modes.

<table>
<thead>
<tr>
<th>SKIP</th>
<th>CHP</th>
<th>FAST</th>
<th>TIME TO FIRST O/P</th>
<th>TIME TO SUBSEQUENTS O/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>24x SF/Fmod</td>
<td>SF/Fmod</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>66x SF/Fmod</td>
<td>3xSF/Fmod</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>3x SF/Fmod</td>
<td>SF/Fmod</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>3x SF/Fmod</td>
<td>3x SF/Fmod</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3x SF/Fmod</td>
<td>SF/Fmod</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>6x SF/Fmod</td>
<td>3x SF/Fmod</td>
</tr>
</tbody>
</table>

Table No.4.1
Settling Time and Throughput Rate
for all the possible modes (X=1 or 0)

The Update Rate and Noise Figure in $\mu$V against all the possible Input Ranges, for all possible values of SF word (CHP=0 and SKIP=1) are outlined in Table 4.2 [9]. The output Peak-to-Peak Resolution versus Input Ranges for all possible SF words is outlined in Table No.4.3 [9].
<table>
<thead>
<tr>
<th>OUTPUT DATA RATE</th>
<th>-3Db FREQ Hz</th>
<th>SF WORD</th>
<th>SETTLE TIME ms</th>
<th>+/-1.28 V</th>
<th>+/-640 mV</th>
<th>+/-320 mV</th>
<th>+/-160 mV</th>
<th>+/-80 mV</th>
<th>+/-40 mV</th>
<th>+/-20 mV</th>
</tr>
</thead>
<tbody>
<tr>
<td>150Hz</td>
<td>39.3</td>
<td>2048</td>
<td>20</td>
<td>2.6</td>
<td>1.45</td>
<td>0.87</td>
<td>0.6</td>
<td>0.43</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>200Hz</td>
<td>52.4</td>
<td>1536</td>
<td>15</td>
<td>3.0</td>
<td>1.66</td>
<td>1.02</td>
<td>0.69</td>
<td>0.48</td>
<td>0.32</td>
<td>0.22</td>
</tr>
<tr>
<td>300Hz</td>
<td>78.6</td>
<td>1024</td>
<td>10</td>
<td>3.7</td>
<td>2.16</td>
<td>0.84</td>
<td>0.58</td>
<td>0.41</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>400Hz</td>
<td>104.8</td>
<td>768</td>
<td>7.5</td>
<td>4.2</td>
<td>2.3</td>
<td>1.46</td>
<td>1.0</td>
<td>0.69</td>
<td>0.46</td>
<td>0.32</td>
</tr>
<tr>
<td>600Hz</td>
<td>157.1</td>
<td>512</td>
<td>5</td>
<td>5.2</td>
<td>2.9</td>
<td>1.78</td>
<td>1.2</td>
<td>0.85</td>
<td>0.58</td>
<td>0.41</td>
</tr>
<tr>
<td>800Hz</td>
<td>209.6</td>
<td>384</td>
<td>6.35</td>
<td>6.6</td>
<td>3.3</td>
<td>2.1</td>
<td>1.4</td>
<td>0.98</td>
<td>0.66</td>
<td>0.47</td>
</tr>
<tr>
<td>1200Hz</td>
<td>314.2</td>
<td>256</td>
<td>2.5</td>
<td>7.8</td>
<td>4.3</td>
<td>2.6</td>
<td>1.8</td>
<td>1.27</td>
<td>0.82</td>
<td>0.57</td>
</tr>
<tr>
<td>1600Hz</td>
<td>419.2</td>
<td>192</td>
<td>1.87</td>
<td>10.9</td>
<td>5.4</td>
<td>3.5</td>
<td>2.18</td>
<td>1.51</td>
<td>0.94</td>
<td>0.64</td>
</tr>
<tr>
<td>2400Hz</td>
<td>629.1</td>
<td>128</td>
<td>1.25</td>
<td>27.1</td>
<td>13.9</td>
<td>7.3</td>
<td>3.5</td>
<td>2.22</td>
<td>1.24</td>
<td>0.83</td>
</tr>
<tr>
<td>3200Hz</td>
<td>838.4</td>
<td>96</td>
<td>0.94</td>
<td>47.4</td>
<td>24.4</td>
<td>11.4</td>
<td>5.3</td>
<td>3.1</td>
<td>1.9</td>
<td>1.0</td>
</tr>
<tr>
<td>4800Hz</td>
<td>1260.1</td>
<td>64</td>
<td>0.625</td>
<td>99.9</td>
<td>50.3</td>
<td>24.5</td>
<td>12.5</td>
<td>6.5</td>
<td>3.3</td>
<td>1.7</td>
</tr>
<tr>
<td>6400Hz</td>
<td>1676.3</td>
<td>48</td>
<td>0.47</td>
<td>193.4</td>
<td>97.4</td>
<td>48.4</td>
<td>24.4</td>
<td>11.8</td>
<td>6.6</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Table 4.2

Output Noise vs. Input Range and Update Rate (CHP = 0, SKIP = 1) Typical
Output RMS Noise in µV

4.3.6 Chop Mode (SKIP = 1, CHP = 1)

With CHP = 1, the AD7731 alternatively reverses the ADC inputs, producing an output which contains the channel offset when not reversed and the negative of the offset when reversed. As a result, when operating in SKIP mode, the user has to take two subsequent outputs from the AD7731 and average them to produce a valid output from the first stage filter. While operating in this mode, gives the benefits of chopping without the longer settling time associated with the 22-tap FIR filter, care should be taken with input signals near positive full-scale or negative full-scale (zero-scale in unipolar mode).
Since the calibration coefficients are generated for the averaged offset, and not for the individual offsets represented in each sample, one of the two samples in the pair may record an all 1s or all 0s reading. If this happens it will result in an error in the averaged reading. Time to first output for the pair is $1/$Output Rate in this mode. However, because should be taken two outputs to derive a correct chopped result, the time to get two outputs for averaging is $2 \times 1/$Output Rate [9]. If is need to use chopping with-out the longer settling time associated with the 22-tap FIR filter, it is recommended that the part be used in FAST Step mode.

4.3.7 Normal FIR Operation (SKIP = 0)

The normal mode of operation of the second stage filter is as a 22-tap low-pass FIR filter. This second stage filter processes the output of the first stage filter and the net frequency response of the filter is simply a product of the filter response of both filters.

<table>
<thead>
<tr>
<th>OUTPUT DATA RATE</th>
<th>-3Db FREQ Hz</th>
<th>SF WORD</th>
<th>SET TIME ms</th>
<th>+/- 1.228V</th>
<th>+/- 640 mV</th>
<th>+/- 320 mV</th>
<th>+/- 160 mV</th>
<th>+/- 80 mV</th>
<th>+/- 40 mV</th>
<th>+/- 20 mV</th>
</tr>
</thead>
<tbody>
<tr>
<td>150Hz</td>
<td>39.3Hz</td>
<td>2048</td>
<td>20ms</td>
<td>17.5</td>
<td>17</td>
<td>17</td>
<td>16.5</td>
<td>16</td>
<td>15.5</td>
<td>15</td>
</tr>
<tr>
<td>200Hz</td>
<td>52.4Hz</td>
<td>1536</td>
<td>15ms</td>
<td>17</td>
<td>17</td>
<td>16.5</td>
<td>16.5</td>
<td>16</td>
<td>15.5</td>
<td>15</td>
</tr>
<tr>
<td>300Hz</td>
<td>78.6Hz</td>
<td>1024</td>
<td>10ms</td>
<td>17</td>
<td>16.5</td>
<td>16.5</td>
<td>16</td>
<td>15.5</td>
<td>15</td>
<td>14.5</td>
</tr>
<tr>
<td>400Hz</td>
<td>104.8Hz</td>
<td>768</td>
<td>7.5ms</td>
<td>16.5</td>
<td>16.5</td>
<td>16</td>
<td>15.5</td>
<td>15</td>
<td>14.5</td>
<td>14</td>
</tr>
<tr>
<td>600Hz</td>
<td>157Hz</td>
<td>512</td>
<td>5ms</td>
<td>16.5</td>
<td>16</td>
<td>16</td>
<td>15.5</td>
<td>15</td>
<td>14.5</td>
<td>14</td>
</tr>
<tr>
<td>800Hz</td>
<td>209.6Hz</td>
<td>384</td>
<td>3.75ms</td>
<td>16</td>
<td>16</td>
<td>15.5</td>
<td>15</td>
<td>14.5</td>
<td>14.5</td>
<td>14</td>
</tr>
<tr>
<td>1200Hz</td>
<td>314Hz</td>
<td>256</td>
<td>2.5ms</td>
<td>15.5</td>
<td>15.5</td>
<td>15.5</td>
<td>15</td>
<td>14.5</td>
<td>14</td>
<td>13.5</td>
</tr>
<tr>
<td>1600Hz</td>
<td>419.2Hz</td>
<td>192</td>
<td>1.87ms</td>
<td>15</td>
<td>15</td>
<td>14.5</td>
<td>14</td>
<td>14</td>
<td>13.5</td>
<td>13</td>
</tr>
<tr>
<td>2400Hz</td>
<td>629Hz</td>
<td>128</td>
<td>1.25ms</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>13.5</td>
<td>13.5</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>3200Hz</td>
<td>838.4Hz</td>
<td>96</td>
<td>0.94ms</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>12.5</td>
</tr>
<tr>
<td>4800Hz</td>
<td>1260Hz</td>
<td>64</td>
<td>0.625ms</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>11.5</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>6400Hz</td>
<td>1676Hz</td>
<td>48</td>
<td>0.47ms</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 4.3
Peak-to-Peak Resolution vs. Input Range and Update Rate (CHP = 0, SKIP =1)
Peak-to-Peak Resolution in Bits
The FIR filter exhibits a very good linearity of the phase but with the expense of a poor stop band. In order to increase the attenuation in stop band the number of taps is increased therefore the settling time is increased accordingly. The overall filter response of the AD7731 is guaranteed to have no overshoot.

4.3.8 Nonchop Mode (SKIP = 0, CHP = 0)

With CHOP mode disabled and SKIP mode disabled, the only function of the second stage filter is to give the overall frequency response. Figure No.4.10 [9] shows the frequency response for the AD7731 with the second stage filter is set for normal FIR operation, chop mode disabled, the decimal equivalent of the word in the SF bits set to 1536 and a master clock frequency of 4.9152 MHz. The response is analogous to that of Figure 4.7 [9] with the three-times larger SF word producing the same 200 Hz output rate. Once again, the response will scale proportionally with master clock frequency.

![Graph](image)

**Figure 4.7**

Full frequency response for SKIP=0,CHP=1,SF=512
The response is shown from dc to 100 Hz. The rejection at 50 Hz ± 1 Hz and 60 Hz ± 1 Hz is better than 88 Db. The -3 dB frequency for the frequency response of the AD7731 with the second stage filter set for normal FIR operation and chop mode enabled is determined by the following relationship: \( f_{3 \text{ dB}} = 0.039 \times f_{\text{MOD}} \times \frac{1}{\text{SF}} \) (CHP = 0) In this case, \( f_{(3\text{dB})} = 7.8 \) Hz and the stop-band, where the attenuation is greater than 64.5 dB, is determined by: \( f_{-\text{STOP}} = 0.14 \times f_{\text{MOD}} \times \frac{1}{\text{SF}} \) (CHP = 0) In this case, \( f_{-\text{Stop}} = 28 \) Hz. Figure No.4.11 [9] shows the frequency response for the same set of conditions as in Figure No 4.8 [9] but in this case the response in shown out to 600 Hz. This plot is comparable to that of Figure 4.7 [9] for the same set of Data.

The most notable difference is the absence of the peaks in the response at 200 Hz and 400 Hz. As a result, interference at these frequencies will be effectively eliminated before being aliased back to dc. Figure 4.10 [9] show the overall characteristic for SKIP=0, CHP=0, SF=1536, and Figure No. 4.11 [9] the expanded full frequency response.

<table>
<thead>
<tr>
<th>Out. Data Rate</th>
<th>-3dB Freq. (Hz)</th>
<th>SF Word</th>
<th>Norm Oper. (ms)</th>
<th>Fast Step (ms)</th>
<th>+/- 1.8 V</th>
<th>+/- 320 mV</th>
<th>+/- 640 mV</th>
<th>+/- 160 mV</th>
<th>+/- 80 mV</th>
<th>+/- 40 mV</th>
<th>+/- 20 mV</th>
</tr>
</thead>
<tbody>
<tr>
<td>50Hz</td>
<td>1.97</td>
<td>2048</td>
<td>440</td>
<td>40</td>
<td>700</td>
<td>425</td>
<td>265</td>
<td>170</td>
<td>120</td>
<td>85</td>
<td>55</td>
</tr>
<tr>
<td>100Hz</td>
<td>3.95</td>
<td>1024</td>
<td>220</td>
<td>20</td>
<td>980</td>
<td>550</td>
<td>330</td>
<td>230</td>
<td>190</td>
<td>115</td>
<td>90</td>
</tr>
<tr>
<td>150Hz</td>
<td>5.92</td>
<td>683</td>
<td>147</td>
<td>13.3</td>
<td>1230</td>
<td>700</td>
<td>445</td>
<td>270</td>
<td>210</td>
<td>140</td>
<td>100</td>
</tr>
<tr>
<td>200Hz</td>
<td>7.9</td>
<td>512</td>
<td>110</td>
<td>10</td>
<td>1260</td>
<td>840</td>
<td>500</td>
<td>340</td>
<td>245</td>
<td>170</td>
<td>105</td>
</tr>
<tr>
<td>400Hz</td>
<td>15.8</td>
<td>256</td>
<td>55</td>
<td>5</td>
<td>2000</td>
<td>1230</td>
<td>690</td>
<td>430</td>
<td>335</td>
<td>215</td>
<td>160</td>
</tr>
<tr>
<td>800Hz</td>
<td>31.6</td>
<td>128</td>
<td>27.5</td>
<td>2.5</td>
<td>3800</td>
<td>2100</td>
<td>1400</td>
<td>760</td>
<td>590</td>
<td>345</td>
<td>220</td>
</tr>
</tbody>
</table>

**Table 4.4**

Output Noise vs. Input Range and Update Rate (CHP = 1, SKIP = 0)

Typical Output RMS Noise in nV.
Figure No. 4.8
Expanded Full frequency response for
SKIP=0, CHP=1, SF=512

Figure 4.9
Overall characteristic for
SKIP=0, CHP=0, SF=1536.
4.3.9 Chop Mode (SKIP = 0, CHP = 1)

With CHOP mode enabled and SKIP mode disabled, the second stage filter is presented with alternating first stage filter outputs and processes data accordingly. It has two primary functions. One is to set the overall frequency response and the second is to eliminate the modulated offset effect, which appears on the output of the first stage filter. Time to first output is $22 \times 1/$Output Rate in this mode. Table 4.4 [9] and Table 4.5 [9] outline the peak-to-peak resolution and typical noise for different setting of SF word.

Figure 4.7 [9] shows the full frequency response of the AD7731 when the second stage filter is set for normal FIR operation. This response is for chop mode enabled with the decimal equivalent of the word in the SF bits set to 512 and a master clock frequency of 4.9152 MHz.
The response will scale proportionately with master clock frequency and it's shown from dc to 100 Hz. The rejection at 50 Hz ± 1 Hz and 60 Hz ± 1 Hz is better than 88 dB. The -3 dB frequency for the frequency response of the AD7731 with the second stage filter set for normal FIR operation and chop mode enabled is determined by the following relationship:

$$ f_{3dB} = 0.0395 \times f_{MOD} \times \frac{1}{(3 \times SF)} \ (CHP = 1) $$

In this case, $f_{3dB} = 7.9$ Hz and the stop-band, where the attenuation is greater than 64.5 dB, is determined by:

$$ f_{STOP} = 0.14 \times f_{MOD} \times \frac{1}{(3 \times SF)} \ (CHP = 1) $$

In this case, $f_{STOP} = 28$ Hz.

**Figure No. 9** [9] shows the frequency response for the same set of conditions as for **Figure No. 8** [9] but in this case the response in shown out to 600 Hz.

<table>
<thead>
<tr>
<th>Output Data Freq.</th>
<th>SF Word</th>
<th>Settling Time Normal</th>
<th>Settling Time Fast</th>
<th>+/- 1.28 V</th>
<th>+/- 640 mV</th>
<th>+/- 320 mV</th>
<th>+/- 160 mV</th>
<th>+/- 80 mV</th>
<th>+/- 40 mV</th>
<th>+/- 20 mV</th>
</tr>
</thead>
<tbody>
<tr>
<td>50Hz 1.97 Hz</td>
<td>2048</td>
<td>440 ms</td>
<td>40 ms</td>
<td>19</td>
<td>19</td>
<td>18.5</td>
<td>18.5</td>
<td>18</td>
<td>17.5</td>
<td>17</td>
</tr>
<tr>
<td>100Hz 3.95 Hz</td>
<td>1024</td>
<td>230 ms</td>
<td>30 ms</td>
<td>19</td>
<td>18.5</td>
<td>18.5</td>
<td>18</td>
<td>17</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>150Hz 5.92 Hz</td>
<td>683</td>
<td>147 ms</td>
<td>13.3 ms</td>
<td>18.5</td>
<td>18</td>
<td>18</td>
<td>17.5</td>
<td>17</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>200Hz 7.9 Hz</td>
<td>512</td>
<td>110 ms</td>
<td>10 ms</td>
<td>18.5</td>
<td>18</td>
<td>17.5</td>
<td>17.5</td>
<td>17</td>
<td>16.5</td>
<td>16</td>
</tr>
<tr>
<td>400Hz 15.8 Hz</td>
<td>256</td>
<td>55 ms</td>
<td>5 ms</td>
<td>17.5</td>
<td>17.5</td>
<td>17</td>
<td>17</td>
<td>16.5</td>
<td>16</td>
<td>15.5</td>
</tr>
<tr>
<td>800Hz 31.6 Hz</td>
<td>128</td>
<td>27.5 ms</td>
<td>2.5 ms</td>
<td>17</td>
<td>16.5</td>
<td>16</td>
<td>16</td>
<td>15.5</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

**Table 4.5**

**Peak-to-Peak Resolution vs. Input Range and Update Rate**

(CHP = 1, SKIP = 0)

This response shows that the attenuation of input frequencies close to 200 Hz and 400 Hz is significantly less than at other input frequencies. These "peaks" in the frequency response are a by-product of the chopping of the input. The plot of **Figure 4.8** [9] is the amplitude for different input frequencies. Note that because the output rate is
200 Hz for the conditions under which Figure 4.8 is plotted, if something existed in the input frequency domain at 200 Hz, it would be aliased and appear in the output frequency domain at dc. Because of this effect, care should be taken in choosing an output rate, which is close to the line frequency in the application.

For example, if the line frequency is 50 Hz, an output update rate of 50 Hz should not be chosen as it will significantly reduce the AD7731's line frequency rejection (the 50 Hz will appear as a dc component with only 6 dB attenuation). However, choosing 60 Hz as the output rate (SF = 1707) will give better than 90 dB attenuation of the aliased line frequency. In a similar fashion, if the line frequency is 60 Hz, it is recommended that the user choose an output update rate of 50 Hz (SF = 2048).

4.3.10 FAST Step Mode (SKIP = 0, FAST = 1)

Second mode of operation of the second stage filter is in FAST Step mode, which enables it to respond rapidly to step inputs even when the second stage filter is in the loop. The FAST Step mode isn’t relevant with SKIP mode enabled.
The FAST Step mode is enabled by placing a 1 in the FAST bit of the Filter Register. If the FAST bit is 0, the part continues to process step inputs with the normal FIR filter as the second stage filter. With FAST Step mode enabled, the second stage filter will continue to process steady state inputs with the filter in its normal FIR mode of operation. However, the part is continuously monitoring the output of the first stage filter and comparing it with the second-previous output. If the difference between these two outputs is greater than a predetermined threshold (1% of full scale), the second stage filter switches to a simple moving average computation. This also happens when a change in channels takes place regardless of how close the voltages on the two channels are. When the change is detected, the STDY bit of the Status Register goes to 1. The initial number of averages in the moving average computation is either 2 (chop enabled)
or 1 (chop disabled). The number of averages will be held at this value as long as the
threshold is exceeded. Once the threshold is no longer exceeded (the step on the analog
input has settled), the number of outputs used to compute the moving average output is
increased.

The first and second outputs from the first stage filter where the threshold is no
longer exceeded is computed as an average by 2, then 4 outputs with an average of 4, 8
outputs with an average of 8 and 6 outputs with an average of 16. At this time, the second
stage filter reverts back to its normal FIR mode of operation.

When the second stage filter reverts back to the normal FIR, the STDY bit of the Status
Register goes to 0. In FAST Step mode, the part has settled to the new value much faster.
For example, with CHP = 1, the FAST Step mode settles to its value in two outputs while
the normal mode settling takes 23 outputs.

Between the second and 23rd output, the FAST Step mode produces a settled
result but with additional noise compared to the specified noise level for its operating
conditions. This noise level starts at approximately 3 times the final noise converging to
FIR mode performance. The complete settling time to where the part is back within the
specified noise number is the same for FAST Step mode and for normal mode. Since the
part is synchronized when a channel change takes place, it will not produce an output
until the filter (either FAST Step or FIR) is settled. As can be seen from Table 4.1 [9],
the FAST Step mode gives a much earlier indication of where the output channel is going
and what its new value is. This feature is very useful in scanning multiple channels where
the microcontroller does not have to wait for the FIR settling time to see if a channel has
changed value. In this case, the part can be set up with CHP = 1, SKIP = 0 and FAST = 1.
This takes advantage of the low drift, better noise immunity benefits of the CHOP mode.

When a change in channels takes place, the part enters FAST Step mode and
provides an output result in $2 \times 1/\text{Output Rate}$. Note, if the FAST bit is set and the part
operated in single con-version mode, the AD7731 will continue to output results until the
STDY bit goes to 0. The part was tested for different configurations of the filter register
with all the combination of the SF word in order to find the optimum setting. The results
of this testing is outlined in Chapter 6, which is dedicated to the overall results and
conclusion.
CHAPTER 5

5.1 Software and Programming

The System is under control of two main programs and the third is implemented in the ADC subsystem. First program is implemented in the PC and represent the core of the system. All the commands are initiated from this one and all the data are returned here. The processing of the cold junction, and scaling of the binary data is also executed here. The second main program is implemented in microcontroller and is the switchboard between the PC and the ADC. All the communication from the PC and to PC is implemented using the SCI port facilities.

The SCI (serial communication interface) is a regular RS-232 port implemented on the RxD (Pin PS0) and TxD (Pin PS1). The baud rate is selectable between 110 baud to 38400 baud. For the purpose of speeding the communication the speed was chosen at 38400 baud. The selection is made writing to SC0BH and SC0BL register on the microcontroller. For a Clock rate of 8.0MHz (the board is running with this clock) the baud rate is implemented writing ‘BR’ to the above register. The value for ‘BR’ is obtained if the clock of the microcontroller is divided with 16 and then with the desired baud rate therefore \[ BR = \frac{8000000}{(38400 \times 16)} = 13.02 \]. Table No.5.1 [9] outlined the selection for the desired BAUD rate.

The next function of the microcontroller program is to perform the communication between the microcontroller and the ADC chip. This communication is implemented on the SPI port, and is wired to the pins PS4 to PS7 of this port. This communication is carried out on a high-speed asynchronous transfer, with a transfer rate selectable with the SP0BR register (bits SPR2, SPR1, SPR0).
<table>
<thead>
<tr>
<th>Desired SCI Baud Rate</th>
<th>BR Divisor for P=4.0MHz</th>
<th>BR Divisor for P=8.0 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>2273</td>
<td>4545</td>
</tr>
<tr>
<td>300</td>
<td>833</td>
<td>1667</td>
</tr>
<tr>
<td>600</td>
<td>417</td>
<td>833</td>
</tr>
<tr>
<td>1200</td>
<td>208</td>
<td>417</td>
</tr>
<tr>
<td>2400</td>
<td>104</td>
<td>208</td>
</tr>
<tr>
<td>4800</td>
<td>52</td>
<td>104</td>
</tr>
<tr>
<td>9600</td>
<td>28</td>
<td>52</td>
</tr>
<tr>
<td>14400</td>
<td>17</td>
<td>35</td>
</tr>
<tr>
<td>19200</td>
<td>13</td>
<td>26</td>
</tr>
<tr>
<td>38400</td>
<td>-</td>
<td>13</td>
</tr>
</tbody>
</table>

Table No.5.1
Selection for the BAUD rate of the SCI communication.

The speed was configured for a rate of 31.3kHz for a reason, which will be shown in the conclusion section. Table No.5.2 [9] outlines the selection of clock speed for SPI communication. The reading of the ADC involves also to monitor the SYNC, RDY, RESET signals, which signals are connected to PORTB pins PB2 to PB4 and for the Calibration point of view also PB5 to PB7 is used to control the Calibration Relay.

<table>
<thead>
<tr>
<th>SPR2</th>
<th>SPR1</th>
<th>SPR0</th>
<th>E Clock Divisor</th>
<th>Frequency at E Clock=4MHz</th>
<th>Frequency at E Clock=8MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2.0MHz</td>
<td>4.0MHz</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>1.0MHz</td>
<td>2.0MHz</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>500KHz</td>
<td>1.0MHz</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>16</td>
<td>250KHz</td>
<td>500KHz</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>125KHz</td>
<td>250KHz</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>62.5KHz</td>
<td>125KHz</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>128</td>
<td>31.3KHz</td>
<td>62.5KHz</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>15.6KHz</td>
<td>31.3KHz</td>
</tr>
</tbody>
</table>

Table No.5.2
SPI clock rate selection

The Microcontroller program was written in a 'modular' fashion implementing 'Subroutines' for all the communication function. Comments were added to the most
important segments of the program to help to better understand the functions implemented there. The Main program starts at address 8000H in the Flash EPROM area of the microcontroller. The RESET vector points to this address, therefore the start sequence of the program is doing an ADC Software Reset holding high the line RESET for 50ms. The del50 Subroutine is timing this delay therefore this Subroutine is used extensively all over the program. The PC program conditions the flow of the program, which is the Master Program. The program is expecting an ‘Initiation’ of communication from PC and after this is performed, it’s doing an handshake to establish the connection and is entering a WAIT mode where is expecting a READING FROM ADC REGISTERS command from PC. After a task is executed the program is returning to WAIT mode and is expecting a new command.

The set of commands and the flow of the program are described in the PC Visual Basic Graphic User Interface.

The software for the microcontroller was developed in Assembly Language of the Motorola 68 Family. The assembler used is a version of Demo Software supplied with the book ‘EMBEDDED MICROCOMPUTER SYSTEM, Real time Interfacing’ by Jonathan W. Valvano [8]. This publication was referenced otherwise extensively for this project. The CD, which is accompanying this book, is containing beside of the assembler for the Motorola 68HC12 family of microcontrollers, also assemblers for other microcontrollers families from Motorola (68HC05, 68HC11, 68HC16).

The assembler is part of a larger package of software, which also features a very original simulator for real time applications, where a cod written in this environment in assembly language could be ‘run’ on the computer, not being necessary to load this in the microcontroller memory.

This program could simulate inputs and outputs, timers and counters which help to debug cod written for digital I/O and for the Timer and Counter subsystem of the microcontroller. For the communication ports still there is not possible to use the ‘Simulator’ for debug because of the high speed of the events and the impossibility to capture ‘Register’ when the cod is running. Thereafter this type of debugging is possible to be carried out using a ‘Logic Analyzer’.
To solve the communication problems it was also used for debugging another software ‘StingRay’, which is a real-time environment used to run the microcontroller applications under its command and to do debugging while the microcontroller are running.

This software is using BDM (Background Debug Mode) to spy the Inside of during the time when this is running. This is possible due to a special pin which this microcontroller is equipped with and using this interface, it is possible to read Register and Memory locations in the time the program is executed.

The BDM is using the time when the Accumulator is doing other type of operations but Memory fetch or Register manipulation, to read the contents of the Registers and Memory locations. The reading from the microcontroller is send to the PC via BDM pin. This pin is connected to a special device, SDI (Serial Debug Interface) to communicate to PC. This Software despite of the lower price is really useful. All kind of debugging could be carried out with this software tool. The software is a product of ‘SID PRICE’, an Internet based company. The SDI is provided by another Internet based company ‘KEVIN ROSS’. Both companies are listed on the MiniIDE site.

5.2 Graphic User Interface Description

The PC software was written in Visual Basic and is the Interface between the System and the user. From here the commands to the System are send and all the readings from the System are returned. These readings could be Status Registers or Data Registers. Beside of communication with the microcontroller there is implemented in this program routines for Data Processing necessary to carry out the tasks for this application. A description and a flow chart of this program will make more understandable all the operation that this System performs.

Once MCUProject.exe is running, the user is presented with a large canvas consisting of 4 panels: Command, Calibration, Output and Registers panels are aligned to the top, right, bottom and left panels respectively. Panel Descriptions:
**Top or Command Panel:** used to initiate all tasks (different color per task).

**Right or Calibration Panel:** these are only displayed after a Thermo or Cold Calibration (displayed in cyan).

**Bottom or Output Panel:** displayed after Mode 1 or 2 Data Retrieval.

**Left or the Registers Panel:** displayed only once after Read Registers (displayed in yellow).

The operations performed from the Front Panel and the flow of the program are described next:

1) MCU part is turned on
2) MCUProj.exe is running
3) “Connect” Button is clicked
4) The red rectangle changes colors from red to yellow to green. If it remains on yellow for a long time, a connection error will occur.
5) The user is only allowed to read registers once the software is online.
6) When the “Read Registers” is clicked, the function Get Input (19) is called. This function sends “RG” to the Microcontroller (MCU) via the Serial Port and waits until at least 19 bytes of data arrives in the serial port buffer. Once the buffer has enough bytes it dumps the buffer into the global variable ‘the Input’ defined as String.
7) My Class is an instance created by the class Sort Data. Sort Data takes the Input and filters all the values. An Example of the Input would be “st*mdゥфтゥmdゥgnsOd”. The key flags are in bold.
8) Data comes in as ASCII and needs to be converted into decimal once extracted from ‘the Input’.
9) In the example above the Status Register carries only one byte: This is converted with the Predefined VB Function Asc (variable) that converts the ASCII character to its decimal value, thus Asc (“*”)=42. This number is further converted into binary by means of the user-defined function bin (integer).
10) Status, Mode, Gain, Filter and Offset are written the same way.
11) Cold Calibration is done by opening another Canvas (Form1), which is controlled by a timer and a progress bar.
12) A timer is as a control that calls a function at equal intervals repeatedly. In the program the timer adds another percent to the progress bar and checks to see if it has reached maximum value. When the timer reaches a certain value it sends “cm” and filter values to the MCU. Therefore the value for the Filter Register could be ‘inputted’ from the Front Panel writing to the Filter Register Box or the Default value for this register will be overwritten by default when the type of Calibration is choused. When maximum value is reached the incoming data is converted from ASCII to binary and inserted into the Calibration Panel Boxes. (Right aligned panel on the main canvas).

13) After a Cold Calibration user should click on “Cold100” which extracts 100 cold calibration values and averages them out. This button opens another Form (C100). This form automatically sends “cd” to the MCU and receives 100 bytes of Cold Junction Data. This data is averaged and put into a global variable called dblCold defined as a floating-point data.

14) Almost the same procedure for Thermocouple calibration but instead of Form1 it opens Form2 that has a copy of the same functions as Form1 with the exception of the flag “cm” which is replaced by “cp”. No “Cold100” to be done.

15) If either Mode1 or Mode 2 buttons are pressed the Filter Boxes changes values accordingly.

16) Retrieve Data does nothing more then call function get data.

17) The function get data sends the flag “dt” and waits for the data to arrive. Once arrived it places it in “theInput” variable.

18) “TheInput” is processed and displayed in the Output panel (aligned to the bottom of the canvas)

19) To Disconnect press on the same red button used to connect. This sends “rs” to the MCU, which tells it to reset itself. The ‘Disconnect’ button should be pressed before turning the MCU off.
For the most part of the program the Visual Basic Predefined Function where sufficient, but despite of the rich Function Libraries, which reside in the Software was necessary to add few new \textit{User Defined Functions}.

\section*{5.3 User Defined Functions:}

\subsection*{5.3.1 Get Input}

\textit{Description:} Waits until the Serial Port Buffer length (bytes) is greater than max (integer).

\section*{5.4 Function Bin}

\textit{Description:} Converts a decimal number to binary form. (dec as integer)

\subsection*{5.4.1 Function get data}

\textit{Description:} Function split into two parts, model and mode2.

If it’s mode 1 it is specifically made for Cold Calibration.

If it’s mode 2 it is specifically made for Thermocouple Calibration.

(do not take any variables).

\subsection*{5.4.2 Function EnableDisableButtons (Boolean)}

\textit{Description:} Called at different intervals with a Boolean value that could be either True or False. If True then the all the buttons in the Command Panel are enabled else all are disabled. Doesn’t apply to Connect/Disconnect and Read Register buttons.

\subsection*{5.4.3 Function dec24/16(String)}

\textit{Description:} Takes a string of binary data i.e. “01001110” and turns it into decimal.

\subsection*{5.4.4 Function AddOfficeBorder(hWnd)}

\textit{Description:} Window API Functions to Bevel Edit Boxes the same way they look in MS Office 2000.
CHAPTER 6

6.1 Design Procedure and Testing.

The design of the data acquisition was carried out step by step with testing every stage and then doing the development for the next one. To ensure a better debug and test of the different tasks of the application a few tools were added specially created for the testing purposes.

The first stage of the design the communication with the PC was established. For this a 5X5 keypad was added to the board and the first communication between the Microcontroller and PC tested using this. Port A and Pin 1,0 of port B were configured to do this.

The serial communication testing uses HyperTerminal from the File Application Menu of the PC. The baud rate and the settings for the communication board were set for 9600,8,1 the standard settings for serial communication. This testing was necessary just to establish communication for the first time and not to draw conclusions from the point of view of the speed of communication.

After achieving the first stage another tool a small "logic analyzer", made from a single chip N555 and used to test presence of TTL logic (0, 5 volts), or a train of pulses under a range of 50 KHz was implemented in the system. This tool help to debug status of the different lines between MCU and ADC as well with PC.

After the communication with PC was established, the next stage target to establish communications between the microcontroller and analog/digital converter. This type of communications involves high-speed synchronous transfer. In order to achieve synchronization between the Command Word and the Reading of the Registers or Data, the RDY line was monitored in the MCU program. Because of the speed of the scan in the MCU, there is the possibility that, RDY line could be trapped on the low level before the RDY to be asserted by the command sent from the MCU. Thus the low level will be
interpreted by the MCU like the data that has available to be read from ADC that is not real. To overcome this problem a 0.1 millisecond delay was inserted in the program between the command and the actual reading. Every reading the ADC will be delayed about 100 microseconds; this problem could be overcome if the reading of the ADC would be observed with a real Logic Analyzer and the delay routine tuned for the minimum necessary delay. Therefore, by doing this, the reading from the ADC will speed up and the Output Rate will also. The reading from the ADC will update three data registers in the MCU.

After one reading from the ADC was completed these three data registers will be moved on through the SCI to the PC. If the slow SCI communication will be replaced with a parallel communication the speed of reading the ADC will be increased tremendously.

Another disadvantage of using the slow transfer through the SCI is that the data processing will not be possible to be carried out in the MCU. If the processing would be carried out in the MCU, the speed of the reading of the ADC would decrease further. This was a reason in the PC instead of the MCU. The second reason was the imprecision of the operation in the MCU due to 16-bit operation. In order to increase precision the result would be read from one register and the rest from another register, further having the result concatenated. This will make multiple loops just for carrying out one operation that decreases speed of transfer from the ADC as well.

One possible overcome of this problem will be to use instead of microcontroller a DSP. The DSP has three processors and while one processor is carrying out the communications with the ADC, another that is specialized in floating point operations (math) with high speed will carry out all the processing and the third will compress the result and send it to the PC via Parallel port. This option would make possible converting high-speed signals such as Ultrasound or Vibrations. The complexity of the DSP allows the system to become thoroughly portable.

From the point of view of calibration a few problems can be mentioned about the Cold junction channel. First the calibration for this channel was done with a zero scale and full-scale system. For the zero scale a short between pin 5 and 6 (GND) was done. For the full scale system because of the availability of the just 1.228 high precision
voltage reference instead of the 1.28 V - which is the full scale level, a correction over the signal was necessary to be added after the testing was done over channel 3 and compared with the level from the input of the channel read with four digit external voltmeter. Consequently, a non-linear factor should be added to correct the signal. This nonlinearity appears due to the difference in the full-scale calibration and is reflected like a gain error. After the correction, the signal landed with three decimal equal with the one measured with the external voltmeter.

Another problem related with channel number 1, which is the thermocouple channel, was observed when the input was held on 0 V and the conversion was carried out repeatedly. Instead to receive 24-bits of constant signal, the last 7 bits was changing (flickering). This is because of the noise not properly rejected by the system. For a level of 80 mV this noise represents 60 microvolts.

Due to the compensation of the cold junction through the software, the level of signal, which appears to the input of the system when the thermocouple is at the same temperature with the cold junction, will be at 0V.

This will make the reading of the ambient temperature highly erroneous. To overcome this problem a hardware compensation for the cold junction will bring the signal in the range far from the noise system. (For the ambient temperature, this level will be about 1100 microvolts that is twenty times higher than the noise level.).

To further eliminate noise from the system it was observed that a drain for the cable shield would be better to be done to the thermocouple side and not to the system side. The reason for this is that if the drain will be done to the analog ground that is the signal ground, the noise from the shield will be induced in the signal line. The other option to drain the cable shield to the digital ground which is a very noisy ground due to the microcontroller bus, the noise from this ground will be inserted in the signal line capacitive from the cable shield. For the thermocouple signal was performed repeatedly measurements using a source of temperature in the range of 0 to 300 degrees Celsius. That was observed that the reading is consistent and repetitive.

To establish the most suitable configuration for the Filter Register was effectuated tests to identify the behavior of the ADC for different ‘Output Rates’. So far the Power line frequency 60Hz is the worst case. Choosing this output rate will allow the line
frequency to come through the system along with the signal; therefore the 'Output data' will be hardly corrupted. The opposite represents the 'Output Rate' of 50 Hz which setting will bring a reject of 120 dB over the Line frequency (60Hz). The setting for the Filter Register, which satisfies this, was found to be for the channel 1: 10000000 for Filter 1 and 00000100 for Filter 2. Therefore this is the default setting for the Calibration Thermo and Data Collect Functions. Doing the same testing there was found the setting for the Cannel 3: 00100000 for Filter 1 and 00000010 for Filter 2. Therefore this is used Default setting For Calibration Cold and Cold 100 function.

Despite of this, it’s necessary to calibrate this channel with an already calibrated system. We expect to do a correction over the signal especially because of the nonlinearity of the thermocouple signal with temperature. Therefore a Chart with a preliminary cooling curve for a sample of 7 readings of 1750 points is added in Appendix D.
APPENDIX A: Thermocouple Type K-Table of Conversion
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
</tr>
</thead>
<tbody>
<tr>
<td>-150</td>
<td>-4.81</td>
<td>-4.92</td>
<td>-5.03</td>
<td>-5.14</td>
<td>-5.24</td>
<td>-5.34</td>
<td>-5.43</td>
<td>-5.52</td>
<td>-5.60</td>
<td>-5.68</td>
</tr>
<tr>
<td>-100</td>
<td>-3.49</td>
<td>-3.64</td>
<td>-3.78</td>
<td>-3.92</td>
<td>-4.06</td>
<td>-4.19</td>
<td>-4.32</td>
<td>-4.46</td>
<td>-4.58</td>
<td>-4.70</td>
</tr>
<tr>
<td>-50</td>
<td>-1.86</td>
<td>-2.03</td>
<td>-2.20</td>
<td>-2.37</td>
<td>-2.54</td>
<td>-2.71</td>
<td>-2.87</td>
<td>-3.03</td>
<td>-3.19</td>
<td>-3.34</td>
</tr>
<tr>
<td>0</td>
<td>0.00</td>
<td>0.19</td>
<td>-0.39</td>
<td>-0.58</td>
<td>-0.77</td>
<td>-0.95</td>
<td>-1.14</td>
<td>-1.32</td>
<td>-1.50</td>
<td>-1.68</td>
</tr>
<tr>
<td>+0</td>
<td>0.00</td>
<td>0.20</td>
<td>0.40</td>
<td>0.60</td>
<td>0.80</td>
<td>1.00</td>
<td>1.20</td>
<td>1.40</td>
<td>1.61</td>
<td>1.81</td>
</tr>
<tr>
<td>50</td>
<td>2.02</td>
<td>2.23</td>
<td>2.43</td>
<td>2.64</td>
<td>2.85</td>
<td>3.05</td>
<td>3.26</td>
<td>3.47</td>
<td>3.68</td>
<td>3.89</td>
</tr>
<tr>
<td>100</td>
<td>4.10</td>
<td>4.31</td>
<td>4.51</td>
<td>4.72</td>
<td>4.92</td>
<td>5.13</td>
<td>5.33</td>
<td>5.53</td>
<td>5.73</td>
<td>5.93</td>
</tr>
<tr>
<td>150</td>
<td>6.13</td>
<td>6.33</td>
<td>6.53</td>
<td>6.73</td>
<td>6.93</td>
<td>7.13</td>
<td>7.33</td>
<td>7.53</td>
<td>7.73</td>
<td>7.93</td>
</tr>
<tr>
<td>200</td>
<td>8.13</td>
<td>8.33</td>
<td>8.54</td>
<td>8.74</td>
<td>8.94</td>
<td>9.14</td>
<td>9.34</td>
<td>9.54</td>
<td>9.74</td>
<td>9.95</td>
</tr>
<tr>
<td>250</td>
<td>10.16</td>
<td>10.36</td>
<td>10.57</td>
<td>10.77</td>
<td>10.98</td>
<td>11.18</td>
<td>11.39</td>
<td>11.59</td>
<td>11.80</td>
<td>12.01</td>
</tr>
<tr>
<td>400</td>
<td>16.40</td>
<td>16.81</td>
<td>17.22</td>
<td>17.63</td>
<td>18.04</td>
<td>18.45</td>
<td>18.86</td>
<td>19.27</td>
<td>19.68</td>
<td>20.09</td>
</tr>
<tr>
<td>450</td>
<td>18.51</td>
<td>18.73</td>
<td>19.15</td>
<td>19.58</td>
<td>20.01</td>
<td>20.44</td>
<td>20.87</td>
<td>21.30</td>
<td>21.73</td>
<td>22.16</td>
</tr>
<tr>
<td>550</td>
<td>22.78</td>
<td>22.99</td>
<td>23.20</td>
<td>23.42</td>
<td>23.63</td>
<td>23.84</td>
<td>24.06</td>
<td>24.27</td>
<td>24.49</td>
<td>24.70</td>
</tr>
<tr>
<td>650</td>
<td>27.03</td>
<td>27.24</td>
<td>27.45</td>
<td>27.66</td>
<td>27.87</td>
<td>28.08</td>
<td>28.29</td>
<td>28.50</td>
<td>28.72</td>
<td>28.93</td>
</tr>
<tr>
<td>700</td>
<td>29.14</td>
<td>29.35</td>
<td>29.56</td>
<td>29.77</td>
<td>29.97</td>
<td>30.18</td>
<td>30.39</td>
<td>30.60</td>
<td>30.81</td>
<td>31.02</td>
</tr>
<tr>
<td>750</td>
<td>31.23</td>
<td>31.44</td>
<td>31.65</td>
<td>31.86</td>
<td>32.06</td>
<td>32.27</td>
<td>32.48</td>
<td>32.68</td>
<td>32.89</td>
<td>33.09</td>
</tr>
<tr>
<td>800</td>
<td>33.30</td>
<td>33.50</td>
<td>33.71</td>
<td>33.91</td>
<td>34.12</td>
<td>34.32</td>
<td>34.53</td>
<td>34.73</td>
<td>34.93</td>
<td>35.14</td>
</tr>
<tr>
<td>850</td>
<td>35.34</td>
<td>35.54</td>
<td>35.75</td>
<td>35.95</td>
<td>36.15</td>
<td>36.35</td>
<td>36.55</td>
<td>36.76</td>
<td>36.96</td>
<td>37.16</td>
</tr>
<tr>
<td>900</td>
<td>37.36</td>
<td>37.56</td>
<td>37.76</td>
<td>37.96</td>
<td>38.16</td>
<td>38.36</td>
<td>38.56</td>
<td>38.76</td>
<td>38.95</td>
<td>39.15</td>
</tr>
<tr>
<td>950</td>
<td>39.35</td>
<td>39.55</td>
<td>39.75</td>
<td>39.94</td>
<td>40.14</td>
<td>40.34</td>
<td>40.53</td>
<td>40.73</td>
<td>40.92</td>
<td>41.12</td>
</tr>
<tr>
<td>1000</td>
<td>41.31</td>
<td>41.51</td>
<td>41.70</td>
<td>41.90</td>
<td>42.09</td>
<td>42.29</td>
<td>42.48</td>
<td>42.67</td>
<td>42.87</td>
<td>43.06</td>
</tr>
<tr>
<td>1050</td>
<td>43.25</td>
<td>43.44</td>
<td>43.63</td>
<td>43.83</td>
<td>44.02</td>
<td>44.21</td>
<td>44.40</td>
<td>44.59</td>
<td>44.78</td>
<td>44.97</td>
</tr>
<tr>
<td>1100</td>
<td>45.16</td>
<td>45.35</td>
<td>45.54</td>
<td>45.73</td>
<td>45.92</td>
<td>46.11</td>
<td>46.29</td>
<td>46.48</td>
<td>46.67</td>
<td>46.85</td>
</tr>
<tr>
<td>1150</td>
<td>47.04</td>
<td>47.23</td>
<td>47.41</td>
<td>47.60</td>
<td>47.78</td>
<td>47.97</td>
<td>48.15</td>
<td>48.34</td>
<td>48.52</td>
<td>48.70</td>
</tr>
<tr>
<td>1200</td>
<td>48.89</td>
<td>49.07</td>
<td>49.25</td>
<td>49.43</td>
<td>49.62</td>
<td>49.80</td>
<td>49.98</td>
<td>50.16</td>
<td>50.34</td>
<td>50.52</td>
</tr>
<tr>
<td>1250</td>
<td>50.69</td>
<td>50.87</td>
<td>51.05</td>
<td>51.23</td>
<td>51.41</td>
<td>51.58</td>
<td>51.76</td>
<td>51.94</td>
<td>52.11</td>
<td>52.29</td>
</tr>
<tr>
<td>1300</td>
<td>52.46</td>
<td>52.64</td>
<td>52.81</td>
<td>52.99</td>
<td>53.16</td>
<td>53.34</td>
<td>53.51</td>
<td>53.68</td>
<td>53.85</td>
<td>54.03</td>
</tr>
<tr>
<td>1350</td>
<td>54.20</td>
<td>54.37</td>
<td>54.54</td>
<td>54.71</td>
<td>54.88</td>
<td>55.05</td>
<td>55.22</td>
<td>55.39</td>
<td>55.56</td>
<td>55.73</td>
</tr>
</tbody>
</table>
APPENDIX B: Block Diagram of Motorola 68HC12 Microcontroller
APPENDIX C: Thermocouple Signal Acquisition Board-Electrical Diagram
APPENDIX D: Data Acquired for Seven Reading Samples of a Cooling Curve
APPENDIX E: Microcontroller Assembly Language Program
SCOBDL equ $00C1
PORTS equ $00D6
DDRS equ $00D7
SCOCR1 equ $00C2
SCOCR2 equ $00C3
SCOSR1 equ $00C4
SCOSR2 equ $00C5
SCODRL equ $00C7
PORTA equ $0000 ;define port a
PORTB equ $0001 ;define port b
DDRA equ $0002 ;define ddra
DDRB equ $0003 ;define ddrb
DDRP equ $0057
PORTP equ $0056
COPCTL equ $0016
SP0CR1 equ $00D0
SP0CR2 equ $00D1
SP0BR equ $00D2
SP0SR equ $00D3
SP0DR equ $00D5
PURDS equ $00DB

keytbl equ $9000 ;define constant var. space at $9000
asciibtbl equ $90ff
wkr equ $9200
kkr equ $9202
uur equ $9204
ppr equ $9206
reg equ $9208
ref equ $920a
fil equ $920c
mod equ $920e
offs equ $9210
gnn equ $9212
res equ $9214
cmpl equ $9216
cmp2 equ $9218
zer equ $921a
ful equ $921c
con equ $921e
cld equ $9220

org $800 ;define origin ram at $800
buf rmb 2 ;assign next 2 bytes to the buffer
pattern rmb 2 ;assign 2 bytes to pattern
cnt rmb 2 ;assign 2 bytes for the position
csn rmb 2 ;assign 2 bytes for spare
cntl rmb 2 ;assign 2 bytes for the sound length
pos0 rmb 1 ;next 8 bytes for
pos1 rmb 1
pos2 rmb 1
pos3 rmb 1
pos4 rmb 1
pos5 rmb 1
pos6 rmb 1
pos7 rmb 1
off33  rmb   1
gain11  rmb   1
gain22  rmb   1
gain33  rmb   1
offch31  rmb   1
offch32  rmb   1
offch33  rmb   1
offch11  rmb   1
offch12  rmb   1
offch13  rmb   1
gainch31  rmb   1
gainch32  rmb   1
gainch33  rmb   1
gainch11  rmb   1
gainch12  rmb   1
gainch13  rmb   1
rd3reg  rmb   1
rd2reg  rmb   1
dummy11  rmb   1
dummy22  rmb   1
dummy33  rmb   1
dumml  rmb   1
dummm2  rmb   1
dummm3  rmb   1
cnt     rmb   1
org    $8000 ; origin program in eeprom at $8000
start  clr   PORTA ; clear register
clr   PORTB
clr   PORTP
clr   buf
clr   buf+1
clr   pattern
clr   pattern+1
clr   cnt
clr   cnt+1
clr   csn
clr   csn+1
clr   cnt1
clr   cni1
main  lds   #$0a00 ; init stack at adr $0a00
movb   #$00,COPCTL ; disable COP
ldaa   #'A'
staa   aa
staa   aa+1
ldaa   #'K'
staa   kkk
staa   kkk+1
ldaa   #'N'
staa   nn
staa   nn+1
ldaa   #'s'
staa   stta
ldaa   #'t'
staa   stta+1
ldaa   #'m'
staa   mdda
ldaa   #'d'

116
staa mdda+l
ldaa #'d'
staa dtta
ldaa #'t'
staa dtta+l
ldaa #'f'
staa flt
ldaa #'t'
staa flt+l
ldaa #'o'
staa offt
ldaa #'f'
staa offt+l
ldaa #'g'
staa gn
ldaa #'n'
staa gn+l

**************************************************************************
ldaa #$10
staa DDRP
ldaa #$f0
staa DDRB
bclr PORTB,#$10 ;reset the ADC 7731
jsr ldel
bset PORTB,#$10 ;cease reset
ldaa #26 ;initiated the SCI
staa SC0BDL ;set baud rate at 9600
ldaa #$00
staa SC0CR1 ;set SCI Control Register 1
ldaa #$0C
staa SC0CR2 ;set SCI control Register 2
sty ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF

**************************************************************************
jsr recc ;rec. 'W','W'
ldy wwr
cpy rcdumm
bne sty

**************************************************************************
jsr recc ;rec. 'K','K'
ldy kkr
cpy rcdumm
bne sty

**************************************************************************
jsr recc ;rec. 'U','U'
ldy uur
cpy rcdumm
bne sty

**************************************************************************
jsr recc ;rec. 'P','P'
ldy ppr
cpy rcdumm
bne sty

**************************************************************************
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF

**************************************************************************
lda a
staa trdumm
lda a+a+1
staa trdumm+1
jsr trann ;send 'A','A'

*****************************************************************************
lda kkk
staa trdumm
lda kkk+1
staa trdumm+1
jsr trann ;send 'K','K'

*****************************************************************************
lda nn
staa trdumm
lda nn+1
staa trdumm+1
jsr trann ;send 'N','N'

*****************************************************************************
*First operation is to perform a reading of the registers
*after the part was reset. The microcontroller will expect
*a read registers command from the PC.'r''g'
*****************************************************************************

styl nop

*****************************************************************************
rec8  brclr SC0SR1,#$20,rec8 ;expect request for read reg. 'r'
ldab SC0DRL
stab rr
lda SC0SR1 ;clear RDRF
lda SC0DRL ;clear RDRF
rec9  brclr SC0SR1,#$20,rec9 ;expect to receive 'g'
lda SC0DRL
staa rr+1
ldx res
cpx rr ;compare first with 'r','s'(reset)
beq start ;if equal go reset
ldy reg ;if not equal compare with 'r','g'
cpy rr
bne styl ;if not equal go back to styl
jsr del1 ;if equal go next

* subroutine to read the ADC register
*****************************************************************************
lda a #$E0 ;set ss\,sck,mosi outputs
staa DDRS
lda a #$07
staa SP0BR ;SET BAUD AT 1MHz

*****************************************************************************
*perform reading the status register
*****************************************************************************
lda a #$54 ;set reading the ADC
staa SP0CR1
lda a #$00
staa SP0CR2
lda SP0SR ;dummy read to clear flags
lda SP0DR ;dummy read to clear flags
lda a #$10 ;set reading the status register
staa SP0DR
poll1207 tst  SPOS
bpl  poll1207
ldaa  SPOS       ;dummy read to clear flags
ldaa  SPODR      ;dummy read to clear flags
ldaa  #$00       ;start clock
staa  SPODR

poll07  tst  SPOS
bpl  poll07
ldaa  SPODR
staa  status
ldaa  SPOS       ;dummy read to clear flags
ldaa  SPODR      ;dummy read to clear flags

*perform reading the mode registers
ldaa  #$12       ;set data for reading mode reg.
staa  rd2reg
jsr  read2reg    ;read the mode reg.
ldaa  dummy11
staa  model1
ldaa  dummy22
staa  mode2

*perform reading the filter registers

*perform reading of the offset registers
ldaa  #$15       ;set data for reading offset reg.
staa  rd3reg
jsr  read3reg    ;read the offset register.
ldaa  dummy11
staa  off11
ldaa  dummy11
staa  off22
ldaa  dummy33
staa  off33

*perform reading of the gain registers

*Next send the 's','t' flag to PC and next will send the value
*of the register status

ldaa stta
staa trdumm
ldaa stta+1
staa trdumm+1
jsr trann ;send 's','t' to PC

ldaa status ;send status register to PC
staa SC0DRL

TR19 brcrl SC05R1,#$80,TR19
ldaa SC05R1 ;clear RDRF
ldaa SC05RL ;clear RDRF

*send to PC 'm','d' flag and value of mode registers

ldaa mdda
staa trdumm
ldaa mdda+1
staa trdumm+1
jsr trann ;send 'm','d' to PC

ldaa model
staa trdumm
ldaa mode2
staa trdumm+1
jsr trann ;send model,mode2 to PC

*send to PC 'f','t' flag and after the value of filter registers

ldaa flt
staa trdumm
ldaa flt+1
staa trdumm+1
jsr trann ;send 'f','t' to PC

ldaa filter1
staa trdumm
ldaa filter2
staa trdumm+1
jsr trann ;send filter1,filter2 to PC

*send to PC 'g', 'n' flag followed by gain value registers

ldaa gn
staa trdumm
ldaa gn+1
staa trdumm+1
jsr trann ;send 'g','n', to PC

ldaa gain1
staa trdumm
ldaa gain2
staa trdumm+1
jsr trann ;send gain1,gain2 to PC

ldaa gain3 ;send gain3 register to PC
staa SC0DRL
TR1232 brclr SC0SR1,#$80,TR1232
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF
*******************************************************************************
*send to PC 'o', 'f' flag followed by the value of offset registers
*******************************************************************************
ldaa offt
staa trdumm
ldaa offt+1
staa trdumm+1
jsr trann ;send 'o','f'to PC
*******************************************************************************
ldaa off1
staa trdumm
ldaa off2
staa trdumm+1
jsr trann ;send offset1,offset2 to PC
*******************************************************************************
ldaa off3
staa SC0DRL
*******************************************************************************
TR2222 brclr SC0SR1,#$80,TR2222
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF
*******************************************************************************
sty2
*******************************************************************************
******read command word from PC
*******************************************************************************
*first command from the PC is to do a system zero calibration
*for the channel #3 where the cold junction sensor is connected
*From PC first will expect a command to refresh the value of
*the registers 'r','f', or to perform a calibration 'c','m'for
*the cold junction or to receive 'c','p' to perform calibration
*for the thermocouple
*******************************************************************************
rec81 brclr SC0SR1,#$20,rec81 ;expect registers 'r'or'c'
ldab SC0DRL
stab rr
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF
*******************************************************************************
rec91 brclr SC0SR1,#$20,rec91 ;expect 'f'or's'or 'm'or 'p'
ldaa SC0DRL
staa rr+1
ldx res
cpx rr ;compare with 'r','s'(reset)
lbqeq start ;if equal go reset
ldy zer
cpy rr
lbqeq zeros
ldx ful
cpx rr
lbqeq fullrd
ldy cld
cpy rr
lbqeq colddt
ldx con
```assembly
 cpx rr
 lbeq conred
 ldy ref
 cpy rr ;compare with 'r','f'(refresh rg.)
 lbeq stydta ;if equal go to refresh the rg.
 ldx cmp2
 cpx rr ;if rr='c','p'
 lbeq cal2 ;go calibration 2
 ldy cmp1
 cpy rr ;compare with 'c','m'
 bne sty2 ;if not equal go back to sty2

*calibration1

 bset PORTB,#$20
 bset PORTB,#$40

**styz1**
 ldaa SC0SR1 ;clear RDRF
 ldaa SC0DRL ;clear RDRF

 jsr recc
 ldy fil
 cpy rcdumm
 bne sty71

 ldaa SC0SR1 ;clear RDRF
 ldaa SC0DRL ;clear RDRF
 rec1177 brcrlr SC0SR1,#$20,rec1177 ;expect to receive filter Rg.
 ldaa SC0DRL
 staa filtercl

 ldaa SC0SR1 ;clear RDRF
 ldaa SC0DRL ;clear RDRF

 rec1199 brcrlr SC0SR1,#$20,rec1199 ;expect to receive filter Rg.
 ldaa SC0DRL
 staa filterc2

 ldaa SC0SR1 ;clear RDRF
 ldaa SC0DRL ;clear RDRF

 jsr ldel ;long delay subroutine
 jsr ldel
 jsr ldel ;long delay subroutine
 ldaa filtercl ;set filter registers
 staa filreg1 ;rate with CHF=1(chopper enabled)
 ldaa filterc2 ;for the setting of the of CH3
 staa filreg2
 jsr filter

*After the mode register for the ADC was updated will perform a
*calibration for the channel 3 for the system zero scale calibration
*Set the PB5,PB6 to activated the Rel 1 and Rel2, consequently to remove
*power to the input of the OP Amplifier and to shunt the input. The calibration

122
*will be performed for the 1.28 V range to achieve full code.

```
ldaa #$d1 ;set mode registers for internal
staa modreg1 ;zero scale calibration of CH3
ldaa #$76
staa modreg2
jsr mode ;send mode to ADC
jsr ldel
```

```
rdt01 brclr PORTB,#04,rdt01
bra rdt01
rd01 ldaa #$15 ;set data for reading offset reg
staa rd3reg
jsr ldel
jsr read3reg ;read the gain registers
ldaa dummy11
staa offch31 ;offset for zero int.cal in offch1
ldaa dummy22
staa offch32 ;offset for zero int.cal in offc2
ldaa dummy33
staa offch33 ;offset for zero int.cal in offch3
```

```
bc1rr PORTB,#$20 ;add power back to the input
bc1rr PORTB,#$40
jsr ldel
ldaa #$51 ;set mode registers for one shoot
staa modreg1 ;reading of CH3
ldaa #$76
staa modreg2
jsr mode ;send mode to ADC
jsr ldel
ldaa #$11
staa cont
jsr data
c1r data1
c1r data2
c1r data3
bset PORTF,#$10
jsr ldel ;long delay subroutine
jsr ldel ;long delay subroutine
jsr ldel
```

*After the calibration command was sent the line
*RDY will be monitored to sense when the calibration
*is done and the value of the offset could be read

```
ldaa #$f1 ;set parameters for system
staa modreg1 ;full scale calibration for CH3
ldaa #$76
staa modreg2
jsr mode
jsr ldel
```

**write to mode register for calibration**

```
rdt17 brclr PORTB,#04,rdt17
bra rdt17
```

123
rd17  ldaa  #$16             ;set data for read gain Rg.
staa  rd3reg
jsr   ldel
jsr   read3reg          ;read the offset registers
ldaa  dummy11
staa  gainch31          ;store the gain of CH3 in gainch31
ldaa  dummy22
staa  gainch32          ;store the gain of CH3 in gainch32
ldaa  dummy33
staa  gainch33          ;store the gain of CH3 in gainch33

*After the full scale calibration command was sent the line
*RDY will be monitored to sense when the calibration
*is done and the value of the gain could be read

******************************************************************************
bclr  DORTP,#$10
jsr   ldel
ldaa  #$51             ;set parameters for system
staa  modreg1          ;full scale calibration for CH3
ldaa  #$76
staa  modreg2
jsr   mode
jsr   ldel
ldaa  #$11
staa  cont
jsr   data
clr   data1
clr   data2
clr   data3

******************************************************************************
ldaa  cmp1
staa  trdumm
ldaa  cmp1+1
staa  trdumm+1         ;send 'c','m' to PC
jsr   trann            ;res. flag 'c','m'

******************************************************************************
ldaa  offch31
staa  trdumm
ldaa  offch32
staa  trdumm+1
jsr   trann
ldaa  offch33
staa  SCODRL

TR1272 brc1r  SCOSR1,#$80,TR1272
ldaa  SCOSR1          ;clear RDRF
ldaa  SCODRL          ;clear RDRF
ldaa  gainch31
staa  trdumm
ldaa  gainch32
staa  trdumm+1
jsr   trann
ldaa  gainch33
staa  SCODRL

TR1772 brc1r  SCOSR1,#$80,TR1772
ldaa  SCOSR1          ;clear RDRF
ldaa  SCODRL          ;clear RDRF

******************************************************************************
lbra   sty2
colddt
styl27  ldaa SC0SR1 ;clear RDRF
       ldaa SC0DRL ;clear RDRF

jsr  recc
ldy  fil
cpy  rcdumm
bne  styl27

lda SC0SR1 ;clear RDRF
lda SC0DRL ;clear RDRF
rec1009 brclr SC0SR1,#$20,rec1009 ;expect to receive filtdt1
       lda SC0DRL
       staa filtdt1

lda SC0SR1 ;clear RDRF
lda SC0DRL ;clear RDRF
rec1070 brclr SC0SR1,#$20,rec1070 ;expect to receive filrdt2
       lda SC0DRL
       staa filrdt2

lda SC0SR1 ;clear RDRF
lda SC0DRL ;clear RDRF

lda filtdt1 ;set filter registers
staa filreg1 ;rate with CHP=1(chopper enabled)
lda filtdt1
staa filreg2
jsr  filter

lda #31 ;set parameters for cont.reading
staa modreg1 ;of CH1 (80mV)
lda #76
staa modreg2
jsr  mode
jsr  ldel
jsr  ldel
lda #11
staa cont
jsr  data
clr  dumm3
lda #64
staa dumm3
rdagain1 ldaa #$31 ;set parameters for cont. reading
staa modreg1 ;of CH3 (1.28V)
lda #76
staa modreg2
jsr  mode
clr  data1
clr  data2
clr  data3
clr  data
jsr  send
```assembly

dec dumm3
bne ragain1
lbra sty2

contrd nop
styl11 ldaa SC0SR1 ;clear RDRF
lda SC0DRL ;clear RDRF

jsr recc
ldy fil
cpy rcdumm
bne styl111

lda SC0SR1 ;clear RDRF
lda SC0DRL ;clear RDRF

recl109 brclr SC0SR1,#$20,recl109 ;expect to receive filtdct1
lda SC0DRL
sta filtdct1

lda SC0SR1 ;clear RDRF
lda SC0DRL ;clear RDRF

recl179 brclr SC0SR1,#$20,recl179 ;expect to receive filtdct2
lda SC0DRL
sta filtdct2

lda SC0SR1 ;clear RDRF
lda SC0DRL ;clear RDRF

lda filtdct1 ;set filter registers for the lowest output
sta filreg1 ;rate with CHP=1(chopper enabled)
lda filtdct2
sta filreg2
jsr filter

lda #$31 ;set parameters for cont. reading
sta modreg1 ;of CH1 (80mV)
lda #$34
sta modreg2
jsr mode
jsr ldel
jsr ldel
lda #$11
sta cont
jsr data

clr dumm1
clr dumm2
lda #$07
sta dumm1

loop1 nop
lda #$ff
sta dumm2
rdagain ldaa #$31 ;set parameters for cont. reading
sta modreg1 ;of CH1 (80mV)
lda #$34
```

126
staa modreg2
jsr mode
clr data1
clr data2
clr data3
jsr data
jsr send
dec dummy2
bne rdagain
dec dummy1
bne loop1
ldaa SPOSР ;dummy read to clear flags
ldaa SPOДР ;dummy read to clear flags
bra sty2

;here will update the value
;of register for reading of the data

fullrd bset PORTP,#$10
bra stydata
zeros bset PORTB,#$20
bset PORTB,#$40
stydata nop
sty41 ldaa SCOSR1 ;clear RDRF
ldaa SCODRL ;clear RDRF

jsr recc
ldy mod
cpy rcdumm
bne sty41

ldaa SCOSR1 ;clear RDRF
ldaa SCODRL ;clear RDRF
recc23 brclr SCOSR1,#$20,recc23 ;expect to receive mode1
ldaa SCODRL
staa model1

ldaa SCOSR1 ;clear RDRF
ldaa SCODRL ;clear RDRF
recc24 brclr SCOSR1,#$20,recc24 ;expect to receive mode2
ldaa SCODRL
staa mode2

sty5 ldaa SCOSR1 ;clear RDRF
ldaa SCODRL ;clear RDRF

jsr recc
ldy fil
cpy rcdumm
bne sty5

ldaa SCOSR1 ;clear RDRF
ldaa SCODRL ;clear RDRF
rec9231 brclr SCOSR1,#$20,rec9231 ;expect to receive filter1
ldaa SCODRL
staa filter1

127
jsr  lddl
jsr  lddl
clr  filreg1
clr  filreg2
clr  modreg1
clr  modreg2
*send filter setting to ADC to establish the preliminary conditions
*for reading the data

ldaa  filter11
staa  filreg1
ldaa  filter22
staa  filreg2
jsr  filter
del  del150

ldaa  model1
staa  modreg1
ldaa  mode22
staa  modreg2
jsr  mode
del  del150
ldaa  #$11
staa  cont
jsr  lddl
jsr  data
ldaa  model1
staa  modreg1
ldaa  mode22
staa  modreg2
jsr  mode
del  del150
ldaa  #$11
staa  cont
jsr  data
bcrl  PORTB, #$20
bcrl  PORTB, #$40
bcrl  PORTP, #$10

ldaa  dttta  ;send 'd' flag
staa  SCODRL
brclr  SC0SR1, #$80, TR97
ldaa  dttta+1  ;send 't' flag
staa  SC0DRL
brclr  SC0SR1, #$80, TR72
ldaa  SC0SR1  ;clear RDRF
ldaa  SC0DRL  ;clear RDRF

ldaa  data1  ;send data1 register to PC
staa  SC0DRL
brclr  SC0SR1, #$80, TR79
ldaa  SC0SR1  ;clear RDRF
ldaa  SC0DRL  ;clear RDRF

ldaa  data2  ;send data2 register to PC
staa  SC0DRL
TR272   brclr  SC0SR1,#$80,TR272
ldaa  SC0SR1 ;clear RDRF
ldaa  SC0DRL ;clear RDRF

*******************************************************************************
ldaa  data3 ;send data3 register to PC
staa  SC0DRL
TR271   brclr  SC0SR1,#$80,TR271
ldaa  SC0SR1 ;clear RDRF
ldaa  SC0DRL ;clear RDRF

*******************************************************************************
lbra  sty2

*******************************************************************************
cal2

*******************************************************************************
sty171  ldaa  SC0SR1 ;clear RDRF
ldaa  SC0DRL ;clear RDRF

*******************************************************************************
jsr  recc
ldy  fil
cpy  rcdumm
bne  sty171

*******************************************************************************
ldaa  SC0SR1 ;clear RDRF
ldaa  SC0DRL ;clear RDRF
rec7171   brclr  SC0SR1,#$20,rec7171 ;expect to receive filter calibration1
ldaa  SC0DRL
staa  filtrc21

*******************************************************************************
ldaa  SC0SR1 ;clear RDRF
ldaa  SC0DRL ;clear RDRF

*******************************************************************************
rec1212   brclr  SC0SR1,#$20,rec1212 ;expect to receive filter Rg.
ldaa  SC0DRL
staa  filtrc22

*******************************************************************************
ldaa  SC0SR1 ;clear RDRF
ldaa  SC0DRL ;clear RDRF

*******************************************************************************
ldaa  filtrc21 ;set filter registers for the lowest output
staa  filreg1 ;rate with CHF=1(chopper enabled)
ldaa  filtrc22 ;for setting of calibration operation of CH3
staa  filreg2
jsr  filter

*******************************************************************************
*After the mode register for the ADC was updated will perform a *calibration for the channel 1 for the internal full scale calibration *The calibration will be performed for the internal 80 mV range for achieving full code.
*******************************************************************************
ldaa  #$b1 ;set mode registers for internal
staa  modreg1 ;full scale calibration of CH1
ldaa  #$34
staa  modreg2
jsr mode ;send mode to ADC
jsr ldel

rdt071 brclr PORTB,#04,rd071
bra rdt071
rd071 ldaa #$16 ;set data for reading gain Rg.
staa rd3reg
jsr ldel
jsr read3reg ;read the gain registers
ldaa dummy11
staa gainch11
ldaa dummy22
staa gainch12
ldaa dummy33
staa gainch13
ldaa #$51 ;set mode registers for reading
staa modreg1 ;one shoot of CH1
ldaa #$34
staa modreg2
jsr mode ;send mode to ADC
jsr ldel
ldaa #$11
staa cont
jsr data
clr data1
clr data2
clr data3
bset PORTB,#$80
jsr ldel ;long delay subroutine
jsr ldel

*After the calibration command was sent the line
*RDY will be monitored to sense when the calibration
*is done and the value of the offset could be read

lda a #$91 ;set parameters for system
staa modreg1 ;zero scale internal cal. for CH1
ldaa #$34
staa modreg2
jsr mode
jsr ldel

write to mode register for calibration

rdt217 brclr PORTB,#04,rd107
bra rdt217
rd107 ldaa #$15 ;set data for offset registers
staa rd3reg
jsr ldel
jsr read3reg
ldaa dummy11
staa offch11
ldaa dummy22
staa offch12
ldaa dummy33
staa offch13

*After the full scale calibration command was sent the line
*RDY will be monitored to sense when the calibration
*is done and the value of the offset could be read

```
bclr PORTB,#$80
jsr 1del
jsr 1del
ldaa #$51 ;set mode registers for reading
staa modreg1 ;one shoot of CH1
ldaa #$34
staa modreg2
jsr mode ;send mode to ADC
jsr 1del
ldaa #$11
staa cont
jsr data
clr data1
clr data2
clr data3
ldaa cmp2
staa trdumm
ldaa cmp2+1
staa trdumm+1 ;send 'c','p' to PC(send back the
jsr trann ;res. of the cal. flag 'c','p'
```

```
ldaa offch11
staa trdumm
ldaa offch12
staa trdumm+1
jsr trann
ldaa offch13
staa SC0DRL

TR1171 brclr SC0SR1,#$80,TR1171
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF
ldaa gainch11
staa trdumm
ldaa gainch12
staa trdumm+1
jsr trann
ldaa gainch13
staa SC0DRL

TR1712 brclr SC0SR1,#$80,TR1712
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF
```

```
lbra sty2
```

```
jsr de150 ; if not zero jsr debounce
bra fkey ; after debounce branch to find the key

next ldaa PORTB ; here test if a key is press: PB0-PB1
anda #$03 ; mask PB2-PB7
beq anyk ; jump to the beginning of the test
jsr de150 ; on key stroke jump to debounce

**************

fkey movb #$18,cnt
ldx #$30 ; load 48 (dec) in X

again ldd keytbl1,x ; load last pattern in acc D
andb #$1f ; mask outputs bits
stab PORTA ; store pattern from acc B to port A
lda PORTB ; load portb in acc A
anda #$03
staa buf ; and transfer in buf
ldab PORTA ; load portb
stab buf+1 ; and transfer to buf+1
ldy keytbl1,x ; load first pattern in Y register
cpy buf ; compare the pattern with buf and buf+1
beq ok ; if match jump to ok
dex ; if not decrement two times X and
dex
bmi gback ; if not match back to keys
dec cnt ; dec cnt, and
bra again ; branch back to load next pattern

ok movb cnt,cnt+1 ; transfer cnt in cnt+1
clr cnt ; clear cnt
ldx cnt ; load in X {double reg} cnt+1
lddd asciitbl1,x ; index cnt+1 to asciitbl1 and load pattern
; acc A
std pattern ; store the pattern in 'pattern'

TR6 brc1r SCOSR1,#$80,TR6
movb #$7f,cnt1 ; init. cnt to set the length of the beep

sound bsct PORTB,#$04 ; then beep
jsr de150
bclr PORTB,#$04
jsr de150
dec cnt1
bne sound

**************

bra anykl

**************

gback lbra keys
sty37 nop

**************: here will stay in loop till key released

anykl ldaa PORTA ; load port A in AccA
anda #$00 ; mask PA0-PA4, test PA5-PA7 if not eq 0
beq nextl ; if zero jump to next
jsr de150 ; if not zero {key press} jsr debounce
bra anykl ; after debounce branch to find the key

next1 ldaa PORTB ; here test if a key press to the PB0-PB1
anda #$03 ; mask PB2-PB7
beq gback ; if the key is rel. go to main to restart
bra anykl

132
; delay subroutine

def50  ldx  #$20
outer  ldy  #$20
inner  dey
bne  inner
dex
bne  outer
rts

def1  ldx  #$04
outer1 ldy  #$ff
inner1 dey
bne  inner1
dex
dex
bne  outer1
rts

; filter
nop
ldaa  #$34 ; reset SPI
staa  SP0CR1
ldaa  #$54 ; set writing to ADC
staa  SP0CR1
ldaa  #$00
staa  SP0CR2
ldaa  SP0SR ; dummy read to clear flags
ldaa  SP0DR ; dummy read to clear flags
ldaa  #$03 ; set writing to filter register
staa  SP0DR
poll27 tsta SP0SR
bpl  poll27
ldaa  SP0SR ; dummy read to clear flags
ldaa  SP0DR ; dummy read to clear flags
staa  SP0DR
poll77 tsta SP0SR
bpl  poll77
ldaa  SP0SR ; dummy read to clear flags
ldaa  SP0DR ; dummy read to clear flags
staa  SP0DR
poll29 tsta SP0SR
bpl  poll29
rts

; mode
nop
ldaa  #$34 ; reset SPI
staa  SP0CR1
ldaa  #$54 ; set writing to ADC
staa  SP0CR1
ldaa  #$00
staa  SP0CR2
ldaa  SP0SR ; dummy read to clear flags
ldaa  SP0DR ; dummy read to clear flags
ldaa  #$02 ; set writing to mode register
staa  SP0DR
poll1127 tst    SP0SR
        bpl    poll1127
        ldaa   SP0SR ;dummy read to clear flags
        ldaa   SP0DR ;dummy read to clear flags
        ldaa   modreg1
        staa   SP0DR
poll1177 tst    SP0SR
        bpl    poll1177
        ldaa   SP0SR ;dummy read to clear flags
        ldaa   SP0DR ;dummy read to clear flags
        ldaa   modreg2
        staa   SP0DR
poll1177 tst    SP0SR
        bpl    poll1177
        rts

******************************************************************************
data    nop
        ldaa   #$34 ;reset ADC
        staa   SP0CR1
        ldaa   #$54 ;set reading the ADC data
        staa   SP0CR1
        ldaa   #$00
        staa   SP0CR2
        ldaa   SP0SR ;dummy read to clear flags
        ldaa   SP0DR ;dummy read to clear flags
        rddt
        brclr   PORTB,#04,read
        bra    rddt
        read   ldaa   cont ;set reading the data register
        staa   SP0DR
poll12227 tst   SP0SR
        bpl    poll12227
        ldaa   SP0SR ;dummy read to clear flags
        ldaa   SP0DR ;dummy read to clear flags
        jsr    de11
        jsr    de11
        ldaa   #$00 ;start clock
        staa   SP0DR
poll1291 tst    SP0SR
        bpl    poll1291
        ldaa   SP0DR
        staa   data1
        ldaa   SP0SR ;dummy read to clear flags
        ldaa   SP0DR ;dummy read to clear flags
        ldaa   #$00 ;start clock
        staa   SP0DR
poll1791 tst    SP0SR
        bpl    poll1791
        ldaa   SP0DR
        staa   data2
        ldaa   SP0SR ;dummy read to clear flags
        ldaa   SP0DR ;dummy read to clear flags
        ldaa   #$00 ;start clock
        staa   SP0DR
poll1790 tst    SP0SR
        bpl    poll1790
        ldaa   SP0DR
        staa   data3
ldaa SP0SR ;dummy read to clear flags
ldaa SP0DR ;dummy read to clear flags
rts

send
ldaa dtta ;send 'd' flag
staa SC0DRL
TR907
brclr SC0SR1,#$80,TR907
ldaa dtta+1 ;send 't' flag
staa SC0DRL
TR702
brclr SC0SR1,#$80,TR702
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF

ldaa data1 ;send data1 register to PC
staa SC0DRL
TR709
brclr SC0SR1,#$80,TR709
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF

ldaa data2 ;send data2 register to PC
staa SC0DRL
TR200
brclr SC0SR1,#$80,TR200
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF

ldaa data3 ;send data3 register to PC
staa SC0DRL
TR202
brclr SC0SR1,#$80,TR202
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF
rts

ldel
ldaa #$ff
seven
jsr de150
jsr de150
jsr de150
jsr de150
jsr de150
de60
bne seven
rts

read2reg
ldaa #$34 ;reset ADC
staa SP0CR1
ldaa #$54 ;set reading the ADC
staa SP0CR1
ldaa #$00
staa SP0CR2
ldaa SP0SR ;dummy read to clear flags
ldaa SP0DR ;dummy read to clear flags
ldaa rd2reg ;set reading the mode register
staa SP0DR
poll1217 tst SP0SR
bpl poll1217
ldaa SPOSR ;dummy read to clear flags
ldaa SPODR ;dummy read to clear flags
rddt1 brcr PORTB,#04,read11
bra rddt1
ldaa #$00 ;start clock
read11 staa SP0DR
poll97 tst SPOSR
bpl poll97
ldaa SP0DR
staa dummy11
ldaa SPOSR ;dummy read to clear flags
ldaa SP0DR ;dummy read to clear flags

**************************************************************************

ldaa #$00 ;start clock
staa SP0DR
poll92 tst SPOSR
bpl poll92
ldaa SP0DR
staa dummy22
ldaa SPOSR ;dummy read to clear flags
ldaa SP0DR ;dummy read to clear flags
ldaa rd3reg ;set reading the three registers
staa SP0DR
poll1227 tst SPOSR
bpl poll1227
jsr ldel
ldaa SPOSR ;dummy read to clear flags
ldaa SP0DR ;dummy read to clear flags
rdof brcr PORTB,#04,read1 ;stay for RDY to deassert.
bra rdof
read1 ldaa #$00 ;start clock
staa SP0DR
poll199 tst SPOSR
bpl poll199
ldaa SP0DR
staa dummy11
ldaa SPOSR ;dummy read to clear flags
ldaa SP0DR ;dummy read to clear flags
ldaa #$00 ;start clock
staa SP0DR
poll119 tst SPOSR
bpl poll119
ldaa SP0DR
staa dummy22
ldaa SPOSR ;dummy read to clear flags

* read3reg

ldaa #$34 ;reset ADC
staa SPOCR1
ldaa #$54 ;set reading the ADC
staa SPOCR1
ldaa #$00
staa SPOCR2
ldaa SPOSR ;dummy read to clear flags
ldaa SP0DR ;dummy read to clear flags
ldaa rd3reg ;set reading the three registers
staa SP0DR

ldaa SP0DR ;dummy read to clear flags
ldaa #$00 ;start clock
staa SP0DR
poll179 tst SP0SR
bpl poll179
ldaa SP0DR
staa dummy33
ldaa SP0SR
ldaa SP0DR
rts
*****************************************************************************
recc
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF
rec
brclr SC0SR1,#$20,rec ;expect to receive first W
ldaa SC0DRL
staa rcdumm
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF
recl
brclr SC0SR1,#$20,recl ;expect to receive second W
ldaa SC0DRL
staa rcdumm+1
rts
*****************************************************************************
trann
nop
ldaa trdumm
staa SC0DRL
TR
brclr SC0SR1,#$80,TR
ldaa trdumm+1
staa SC0DRL
TR1
brclr SC0SR1,#$80,TR1
ldaa SC0SR1 ;clear RDRF
ldaa SC0DRL ;clear RDRF
rts
*****************************************************************************
data area
org keytbl
fdb
$0021,$0041,$0081,$0101,$0201,$0022,$0042,$0082,$0102,$0202
fdb $0024,$0044,$0084,$0104,$0204,$0028,$0048,$0088,$0108
fdb $0208,$0030,$0050,$0090,$0110,$0210
org ascii tbl
dc.b 'd','.','F','E','D','S','*','0','#','C','L','7','6','9','B'
dc.b 'Y','4','5','6','A','T','1','2','3','e'
dc.b wwr
dc.b 'W','W'
dc.b kkr
dc.b 'K','K'
dc.b uur
dc.b 'U','U'
dc.b ppr
dc.b 'P','P'
dc.b reg
dc.b 'R','G'
dc.b ref
dc.b 'r','f'
dc.b fil
APPENDIX F: Visual Basic GUI Source Code
Option Explicit
Private Sub cmdConnDisconn_Click()
Shape1.FillColor = vbYellow
If cmdConnDisconn.Caption = "Connect" Then 'check button label
tmrIfOnline = True
MSComm1.Output = "WWKKUUPP" 'send "Wakeup" string to MCU
Do 'wait for data
    DoEvents
Loop Until MSComm1.InBufferCount >= 2
If MSComm1.Input = "AAKKNN" Then 'wait for akn
tmrIfOnline = False
cmdReadRegs.Enabled = True
ConnStat = True
Shape1.FillColor = vbGreen
cmdConnDisconn.Caption = "Disconnect" 'create disc. button
End If
Else
DataForm.List1.Clear
MSComm1.Output = "rs"
Shape1.FillColor = vbRed
EnableDisableButtons (False)
setFlag = 1
ClearRegs
ConnStat = False
cmdConnDisconn.Caption = "Connect" 'create connect button
End If
End Sub

Private Sub cmdReadFScale_Click()
MSComm1.Output = "fu"
Dim md1, md2, ft1, ft2 As Double
'starts here
modeltext.BackColor = vbYellow
modeltext.Locked = True
mode2text.BackColor = vbYellow
mode2text.Locked = True
filter1text.BackColor = vbYellow
filter1text.Locked = True
filter2text.BackColor = vbYellow
filter2text.Locked = True

md1 = dec(modeltext.Text)
md2 = dec(mode2text.Text)
ft1 = dec(filter1text.Text)
ft2 = dec(filter2text.Text)
MSComm1.Output = "md" & Chr(md1) & Chr(md2) & "ft" & Chr(ft1) & Chr(ft2)
tmrSeekData = True
End Sub

Private Sub cmdReadRegs_Click()
MSComm1.Output = "RG" 'send register flag
Get_Input (19)
Dim MyClass As New SortData
If MyClass.Validate(theInput) = True Then
MyClass.Filter (theInput)
EnableDisableButtons (True)
Else
MsgBox "Failure"
End If
End Sub

Private Sub cmdRead2Scale_Click()
MSComml.Output = "ze"
Dim md1, md2, ft1, ft2 As Double
'starts here
model1text.BackColor = vbYellow
model1text.Locked = True
mode2text.BackColor = vbYellow
mode2text.Locked = True
filter1text.BackColor = vbYellow
filter1text.Locked = True
filter2text.BackColor = vbYellow
filter2text.Locked = True
'send text in ascii format
md1 = dec(model1text.Text)
md2 = dec(mode2text.Text)
ft1 = dec(filter1text.Text)
ft2 = dec(filter2text.Text)
MSComml.Output = "md" & Chr(md1) & Chr(md2) & "ft" & Chr(ft1) & Chr(ft2)
tmrSeekData = True
End Sub

Private Sub cmdRefreshRegs_Click()
Dim md1, md2, ft1, ft2, of1, of2, of3, gn1, gn2, gn3 As Integer
MSComml.Output = "rf" 'send tag for refresh
'lock text inputs and highlight
model1text.BackColor = vbYellow
model1text.Locked = True
mode2text.BackColor = vbYellow
mode2text.Locked = True
filter1text.BackColor = vbYellow
filter1text.Locked = True
filter2text.BackColor = vbYellow
filter2text.Locked = True
'send text in ascii format
md1 = dec(model1text.Text)
md2 = dec(mode2text.Text)
ft1 = dec(filter1text.Text)
ft2 = dec(filter2text.Text)
MSComml.Output = "md" & Chr(md1) & Chr(md2) & "ft" & Chr(ft1) & Chr(ft2)
tmrSeekData = True
End Sub

Private Sub Comml_Connect(ByVal bConnect As Boolean)
Comml.ComPort = "2"
End Sub

Private Sub Command1_Click()
Text2.Text = "10000000"
Text1.Text = "00000010"
Text2.BackColor = vbRed
Text1.BackColor = vbRed
Command5.Enabled = True
Load Form1
Form1.Show
End Sub

Private Sub Command2_Click()
Text2.Text = "10000000"
Text1.Text = "00000100"
Text2.BackColor = vbRed
Text1.BackColor = vbRed
Load Form2
Form2.Show
Command5.Enabled = True
End Sub

Private Sub Command3_Click()
szType = "Model"
modeltext.Text = "01010001"
modeltext.BackColor = vbWhite
mode2text.Text = "01110110"
mode2text.BackColor = vbWhite
End Sub

Private Sub Command4_Click()
szType = "Mode2"
modeltext.Text = "01010001"
modeltext.BackColor = vbWhite
mode2text.Text = "00110100"
mode2text.BackColor = vbWhite
End Sub

Private Sub Command6_Click()
c100.Show
End Sub

Private Sub Form_Terminate()
End
End Sub

Private Sub Command5_Click()
Form3.Show
End Sub

Private Sub Form_Click()
Command5.Enabled = True
AddOfficeBorder (Me.hWnd)
End Sub

Private Sub Form_Load()
End Sub

Private Sub Form_Unload(Cancel As Integer)
If MSComm1.PortOpen = False Then
MSComml.Output = "rs"
MSComml.PortOpen = False
End If
End
End Sub

Private Sub mnuConnect_Click()
Call cmdConnDisconn_Click
End Sub

Private Sub mnuExit_Click()
Call Form_Unload(1)
End Sub

Private Sub SpinButton1_Change()
txtSpinner.Text = SpinButton1.Value
txtSpinner.Refresh
End Sub

Private Sub MDIForm_Load()
setFlag = 1
theInput = "a"
EnableDisableButtons (False)
ConnStat = False
MSComml.PortOpen = True

'open port "com1"
End Sub

Private Sub Text1_Change()
filter2text.Text = Text1.Text
filter2text.BackColor = vbWhite
End Sub

Private Sub Text2_Change()
filter1text.Text = Text2.Text
filter1text.BackColor = vbWhite
End Sub

Private Sub tmrIfOnline_Timer()
'send lag warning
MsgBox "Can't Connect, please Reset MCU", vbCritical, "Connection Status"
tmrIfOnline = False
End Sub

Private Sub tmrSeekData_Timer()
If MSComml.InBufferCount >= 2 Then
theInput = MSComml.Input
tmrSeekData = False
get_data
End If
End Sub

------------------------- EOF -------

MS-DOS Name: c100.frm VBName : c100

Private theArray(0 To 1000) As String
Option Explicit
Function extract_to_list()
Dim pos, total As Double
Dim X, Y As Integer
pos = InStr(1, theInput, "dt", vbTextCompare) 'position of first "dt"
Y = 1  'define counter y as "one"
For X = pos + 2 To Len(theInput) Step 5
    theArray(Y) = Mid(theInput, X, 3)
    List1.AddItem convert(db(theArray(Y)))
    Y = Y + 1
    total = CDbl(List1.List(Y - 2)) + CDbl(total)
Next X
Text1.Text = total / List1.ListCount
dblCold = total / List1.ListCount
End Function

Private Sub Command1_Click()
Unload Me
Me.Hide
End Sub

Private Function db(t24 As String) As String
If Len(t24) = 3 Then
    Dim a, b, c As String
    a = Mid(t24, 1, 1)
    b = Mid(t24, 2, 1)
    c = Mid(t24, 3, 1)
    a = bin(Asc(a))
    b = bin(Asc(b))
    c = bin(Asc(c))
db = a & b & c
End If
End Function

Private Function convert(szVoltageBinary As String) As Double
Dim buffer2, buffer1 As Double
Dim szVoltageDec As Double
    If szVoltageBinary <> "00000000000000000000000000000000" Then
        szVoltageDec = dec24(szVoltageBinary)
    'voltage + temperature find
        buffer1 = ((CDbl(szVoltageDec) * 1.228) / 16777215) + ((0.077 * (13946188) ^ 6) / CDbl(szVoltageDec) ^ 6)
        buffer2 = (buffer1 * 1000 - 1000.272) / 3.664
        convert = CDbl(Mid(buffer2, 1, 4))
    End If
End Function

Private Sub Form_Load()
With myform
    .MSComm1.Output = "cd"
End With
Dim ft1, ft2 As Double
'starts here
    .filter1Text.BackColor = vbYellow
    .filter1Text.Locked = True
    .filter2Text.BackColor = vbYellow
    .filter2Text.Locked = True
ft1 = dec(.filter1Text.Text)
ft2 = dec(.filter2Text.Text)
Private Sub Form_Load()
  Dim tx1, tx2 As Long
  tx2 = dec(myform.Text2.Text)
  tx1 = dec(myform.Text1.Text)
  myform.MSComm1.Output = "cm"
  myform.MSComm1.Output = "ft" & Chr(tx2) & Chr(tx1)
  tmrFindData = True
  myform.Text1.BackColor = vbRed
  myform.Text2.BackColor = vbRed
End Sub

Private Sub Timer7_Timer()
  Dim pos As Long
  Dim a As String
  Dim tx1, tx2 As Long
  ProgressBar1.Value = ProgressBar1.Value - 10
  If ProgressBar1.Value = 1590 Then
    tx2 = dec(myform.Text2.Text)
    tx1 = dec(myform.Text1.Text)
    myform.MSComm1.Output = "cm"
    myform.MSComm1.Output = "ft" & Chr(tx2) & Chr(tx1)
    tmrFindData = True
  End If

  If ProgressBar1.Value = 3000 Then
    Form1.Hide
    pos = InStr(1, theInput, "cp", vbTextCompare) + 2
    myform.offch1.Text = bin(Asc(Mid(theInput, pos, 1))) 'off3
    myform.offch2.Text = bin(Asc(Mid(theInput, pos + 1, 1))) 'off3
    myform.offch3.Text = bin(Asc(Mid(theInput, pos + 2, 1))) 'off3
    myform.gch1.Text = bin(Asc(Mid(theInput, pos + 3, 1))) 'off3
    myform.gch2.Text = bin(Asc(Mid(theInput, pos + 4, 1))) 'off3
    myform.gch3.Text = bin(Asc(Mid(theInput, pos + 5, 1))) 'off3
    myform.offch1.BackColor = vbCyan
    myform.offch2.BackColor = vbCyan
    myform.offch3.BackColor = vbCyan
    myform.gch1.BackColor = vbCyan
    myform.gch2.BackColor = vbCyan
    myform.gch3.BackColor = vbCyan
  End If
End Sub
Unload Form1
End If
End Sub

Private Sub tmrFindData_Timer()
If myform.MSComm1.InBufferCount >= 1 Then
theInput = myform.MSComm1.Input
tmrFindData = False
End If
End Sub

------------------------EOF----------------------

MS-DOS Name: Form2.frm VB Name: Form2

Private Sub Form_Load()
Dim tx1, tx2 As Long
    tx2 = dec(myform.Text2.Text)
    tx1 = dec(myform.Text1.Text)
    myform.MSComm1.Output = "cp"
    myform.MSComm1.Output = "ft" & Chr(tx2) & Chr(tx1)
tmrFindData = True
myform.Text1.BackColor = vbRed
myform.Text2.BackColor = vbRed
End Sub

Private Sub Timer7_Timer()
    Dim pos As Long
    Dim a As String
    ProgressBar1.Value = ProgressBar1.Value + 10
    If ProgressBar1.Value = 2110 Then
        Dim tx1, tx2 As Long
        tx2 = dec(myform.Text2.Text)
        tx1 = dec(myform.Text1.Text)
        myform.MSComm1.Output = "cp"
        myform.MSComm1.Output = "ft" & Chr(tx2) & Chr(tx1)
tmrFindData = True
    End If
    If ProgressBar1.Value = 4150 Then
        Form2.Hide
        pos = InStr(1, theInput, "cp", vbTextCompare) + 2
        myform.Offch11.Text = bin(Asc(Mid(theInput, pos, 1))) 'off3
        myform.Offch12.Text = bin(Asc(Mid(theInput, pos + 1, 1))) 'off3
        myform.Offch13.Text = bin(Asc(Mid(theInput, pos + 2, 1))) 'off3
        myform.Gch11.Text = bin(Asc(Mid(theInput, pos + 3, 1))) 'off3
        myform.Gch12.Text = bin(Asc(Mid(theInput, pos + 4, 1))) 'off3
        myform.Gch13.Text = bin(Asc(Mid(theInput, pos + 5, 1))) 'off3
        myform.Offch11.BackColor = vbCyan
myform.Offch12.BackColor = vbCyan
myform.Offch13.BackColor = vbCyan
myform.Gch11.BackColor = vbCyan
myform.Gch12.BackColor = vbCyan
myform.Gch13.BackColor = vbCyan
Unload Form2
theInput = ""
End If
End Sub
Private Sub tmrFindData_Timer()
If myform.MSComml.InBufferCount >= 1 Then
    theInput = myform.MSComml.Input
    tmrFindData = False
End If
End Sub

MS-DOS Name: Form3.frm  VB-Name: Form3

Option Explicit
Private myArray(1 To 100, 0 To 2826) As Double
Private theArray(1 To 5000) As String
Dim counter As Integer
Private Function send_data_flag()
With myform
    .MSComml.Output = "ct"
    Dim ft1, ft2 As Double
    'starts here
    .filter1text.BackColor = vbYellow
    .filter1text.Locked = True
    .filter2text.BackColor = vbYellow
    .filter2text.Locked = True
    ft1 = dec(.filter1text.Text)
    ft2 = dec(.filter2text.Text)
    .MSComml.Output = "ft" & Chr(ft1) & Chr(ft2)
    With myform
        While (.MSComml.InBufferCount < 8925)
            DoEvents
            Wend
        'enter code once the long data is found
        If InStr(1, theInput, "dt", vbTextCompare) >= 0 Then
            theInput = .MSComml.Input
        End If
        End With
        End With
    End Function
Private Sub cmdGetData_Click()
    Label2.Caption = "0"
    Timer1.Enabled = True
End Sub

Private Sub Command1_Click()
    Open Dir1 & Text2.Text For Output As #1
    Dim Y, z, X As Integer
    X = 0
    While X < List2.SelCount 'loop for all selections
        For Y = 1 To 100 'define the first dim of the array
            If Y <= List2.ListCount Then
                If List2.Selected(Y - 1) = True Then
                    For z = 0 To 1825 ' def the 2nd dim of array
                        Print #1, myArray(List2.List(Y - 1), z) 'write to file 4 bytes
                    Next z
                End If
            End If
        Next Y
    End If
End If
X = X + 1 ' go to new line
Next Y 'loop first dim for all counts
Wend
Close #1
End Sub

Public Function do_time()
Dim aint As Integer
aint = CInt(Label2.Caption)
Dim X As Integer 'first dim of array
For X = 1 To List2.ListCount
If List2.List(X) <> "" Then
myArray(aint, X) = List2.List(X)
End If
Next X
Print "f"
End Function

Private Sub Drive1_Change()
Drive1.Path = Drive1.Drive
End Sub

Private Sub Form_Load()
Text1.Text = SpinButton1.Value
End Sub

Private Sub List2_Click()
Dim a, s As Integer
List3.Clear
For a = 0 To List2.ListCount - 1
If List2.Selected(a) = True Then
For s = 1 To 1800
Text2.Text = "trial" & a + 1 & ".txt"
List3.AddItem myArray(a + 1, s)
Next s
End If
Next a
End Sub

Private Sub SpinButton1_Change()
Text1.Text = SpinButton1.Value
List2.Clear
ProgressBar1.max = SpinButton1.Value
End Sub

Private Sub Text1_Change()
Text1.Text = SpinButton1.Value
End Sub
Private Sub Timer1_Timer()
theInput = ""
send_data_flag

If theInput <> "" Then
Label2.Caption = CInt(Label2.Caption) + 1
ProgressBar1.Value = CInt(Label2.Caption)
List2.AddItem Label2.Caption

148
'extract data /

Dim pos, total As Long
Dim X, Y As Integer
pos = InStr(1, theInput, "dt", vbTextCompare) 'position of first "dt"
Y = 1 'define counter y as "one"
For X = pos + 2 To Len(theInput) Step 5
    theArray(Y) = Mid(theInput, X, 3)
    myArray(CInt(Label2.Caption), Y) = Cactus(theArray(Y))
    Y = Y + 1
Next X
'extract data

If CInt(Label2.Caption) < CInt(SpinButton1.Value) Then
    Timer1.Enabled = True
Else
    Timer1.Enabled = False
End If
End If
End Sub

Private Function get_data()
Dim counter As Integer
While (CInt(Label2.Caption) <= SpinButton1.Value)
    Dim pos, total As Long
    Dim X, Y As Integer
    pos = InStr(1, theInput, "dt", vbTextCompare) 'position of first "dt"
    Y = 1
    'define counter y as "one"
    For X = pos + 2 To Len(theInput) Step 5
        theArray(Y) = Mid(theInput, X, 3)
        Y = Y + 1
        DataForm.List1.AddItem Cactus(theArray(Y - 1)) & " +C"
        List2.AddItem Cactus(theArray(Y - 1))
    Next X
    do_time
    If SpinButton1.Value = 1 Then
        List1.AddItem "Reading #" & 1
    Else
        List1.AddItem "Reading #" & Label2.Caption
    End If
    List2.Clear
    theInput = ""
    Label2.Caption = CInt(Label2.Caption) + 1
    Y = 1
    If CInt(Label2.Caption) < SpinButton1.Value Then
        send_data_flag
    Else
        Label2.Caption = CInt(Label2.Caption) + 1
    End If
    Wend
    Label2.Caption = "1"
    DataForm.Show
End Function

---------------------- EOF ------
MS-DOS Name: VB Name:

Global Registers(0 To 14) As String
Global theInput As String
Global bintodec(0 To 7) As String
Global szVoltageBinary As String
Global szVoltageDec As Long
Global szType As String
Global ConnStat As Boolean
Global bintodec24(0 To 23) As String
Global setFlag As Integer
Global dblCold As Double

Global coldavg As Double

Global bintodec16(0 To 15) As String
Global deci(0 To 7) As Long

Public Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hWnd As Long, ByVal nIndex As Long) As Long
Public Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hWnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
Public Declare Function SetWindowPos Lib "user32" (ByVal hWnd As Long, ByVal hWndInsertAfter As Long, ByVal X As Long, ByVal Y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long

Private Const GWL_EXSTYLE = (-20)
Private Const WS_EX_CLIENTEDGE = &H2000
Private Const WS_EX_STATICEDGE = &H20000

Private Const SWP_FRAMECHANGED = &H20
Private Const SWP_NOMOVE = &H2
Private Const SWP_NOOWNERZORDER = &H200
Private Const SWP_NOSIZE = &H1

Private Const SWP_NOZORDER = &H4

Public Function AddOfficeBorder(ByVal hWnd As Long)
Dim lngRetVal As Long
lngRetVal = GetWindowLong(hWnd, GWL_EXSTYLE) + SWP_FRAMECHANGED
SetWindowLong hWnd, 0, 0, 0, 0, lngRetVal
End Function

Function Unscramble_Data()
Dim pos As Integer
pos = InStr(1, theInput, "dt", vbTextCompare) + 2
Dim contents As String

For X = pos To Len(contents) - 2 Step 2
contents = Mid(theInput, X, 2)
End Function
myform.List1.AddItem Cactus(contents)
Next X

End Function
Function Get_Input(max As Integer)
Do
DoEvents
Loop Until myform.MSComm1.InBufferCount >= max
theInput = myform.MSComm1.Input

End Function
Function SetTheRegs()
Dim szColor As Variant
With myform
    .statustext.Text = Registers(0)
    .modeltext.Text = Registers(1)        'setting register
    .mode2text.Text = Registers(2)
    .filter1text.Text = Registers(3)
    .filter2text.Text = Registers(4)
    .gain1text.Text = Registers(5)
    .gain2text.Text = Registers(6)
    .gain3text.Text = Registers(7)
    .offset1text.Text = Registers(8)
    .offset2text.Text = Registers(9)
    .offset3text.Text = Registers(10)
    .Text2.Text = Registers(3)
    .Text1.Text = Registers(4)
If ConnStat = True Then
    szColor = vbYellow
Else
    szColor = vbBlack
End If
AddOfficeBorder (.modeltext.hWnd)
AddOfficeBorder (.mode2text.hWnd)
AddOfficeBorder (.filter1text.hWnd)
AddOfficeBorder (.filter2text.hWnd)
AddOfficeBorder (.offset1text.hWnd)
AddOfficeBorder (.offset2text.hWnd)
AddOfficeBorder (.offset3text.hWnd)
    .modeltext.BackColor = szColor       'highlighting
    .mode2text.BackColor = szColor
    .filter1text.BackColor = szColor
    .filter2text.BackColor = szColor
    .offset1text.BackColor = szColor
    .offset2text.BackColor = szColor
    .offset3text.BackColor = szColor
    .statustext.BackColor = szColor
    .gain1text.BackColor = szColor
    .gain2text.BackColor = szColor
    .gain3text.BackColor = szColor
If szColor = vbYellow Then szColor = Red
    .Text2.BackColor = szColor
    .Text1.BackColor = szColor
End With
End Function
Function bin(dec As Integer)
Dim binar As String  'converts Decimal->Binary
Dim rest As Long
rest = dec
While rest > 0
  binar = rest Mod 2 & binar
  rest = Int(rest / 2)
Wend

Dim X As Integer  'adds 0s as prefix
While Len(binar) < 8
  binar = "0" & binar
Wend
bin = binar
End Function

Function dec(twobin As String)
Dim l As Integer
Dim total As Integer
Dim t As Integer
For l = 0 To 7
  bintodec(7 - l) = Mid(twobin, l + 1, 1)
  bintodec(7 - l) = CInt(bintodec(7 - l)) * (2 ^ (7 - l))
Next l
For t = 0 To 7
  total = bintodec(7 - t) + total
Next t
dec = total
End Function

Function EnableDisableButtons(bOption As Boolean)
With myform
  .cmdReadFScale.Enabled = bOption
  .cmdReadRegs.Enabled = bOption
  .cmdReadZScale.Enabled = bOption
  .cmdRefreshRegs.Enabled = bOption
  .Command1.Enabled = bOption
  .Command2.Enabled = bOption
  .Command3.Enabled = bOption
  .Command4.Enabled = bOption
  .Command6.Enabled = bOption
End With
End Function

Function ClearRegs()
ConnStat = False
Dim i As Integer
For i = 0 To 10
  Registers(i) = ""
Next i
End Function
End With

SetTheRegs

End Function

Function dec16(twobin As String)
    Dim l As Integer
    Dim total As Long
    Dim t As Integer
    For l = 0 To 15
        bintodec16(15 - l) = Mid(twobin, l + 1, 1)
bintodec16(15 - l) = CInt(bintodec16(15 - l)) * (2 ^ (15 - l))
Next l
For t = 0 To 15
    total = bintodec16(15 - t) + total
Next t
dec16 = total
End Function

Function dec24(tobin As String)
    Dim 1 As Integer
    Dim total As Long
    Dim t As Integer
    For l = 0 To 23
        bintodec24(23 - l) = Mid(tobin, 1 + l, 1)
        bintodec24(23 - l) = CInt(bintodec24(23 - l)) * (2 ^ (23 - l))
    Next l
    For t = 0 To 23
        total = bintodec24(23 - t) + total
    Next t
dec24 = total
End Function

Function get_data()
    Dim pos As Integer
    Dim buffer1, buffer2 As Double
    If szType = "Model" Then
        myform.data1.text.BackColor = vbGreen
        myform.data2.text.BackColor = vbGreen
        myform.data3.text.BackColor = vbGreen
        pos = InStr(1, theInput, "dt", vbTextCompare) + 2  'filtering of regs
        myform.data1.text = bin(Asc(Mid(theInput, pos, 1)))  'model
        myform.data2.text = bin(Hex(Mid(theInput, pos + 1, 1)))  'model
        myform.data3.text = bin(Hex(Mid(theInput, pos + 2, 1)))  'model
        szVoltageBinary = myform.data1.text & myform.data2.text & myform.data3.text
        'goes till here
        If szVoltageBinary = "00000000000000000000000000000000" Then
            myform.lblVoltage = "0000.00 mV"
        Else
            szVoltageDec = dec24(szVoltageBinary)
            'voltage + temperature find
            buffer1 = ((CDbl(szVoltageDec) * 1.228) / 16777215) + (0.077
            (13946188) ^ 6) / CDbl(szVoltageDec) ^ 6)
            myform.lblVoltage.Caption = Mid(buffer1, 1, 6) & " V"
        End If
        buffer2 = (buffer1 * 1000 - 1000.272) / 3.664
        myform.lblTemper.Caption = Mid(buffer2, 1, 4) & " °C"
' End If
' for mode 2
If szType = "Mode2" Then
myform.data1ch1.BackColor = vbGreen
myform.data2ch1.BackColor = vbGreen
myform.data3ch1.BackColor = vbGreen
pos = InStr(1, theInput, "dt", vbTextCompare) + 2 "filtering of regs
myform.data1ch1.Text = bin(Asc(Mid(theInput, pos, 1))) "model
myform.data2ch1.Text = bin(Asc(Mid(theInput, pos + 1, 1))) "model
myform.data3ch1.Text = bin(Asc(Mid(theInput, pos + 2, 1))) "model
szVoltageBinary = myform.data1ch1.Text & myform.data2ch1.Text &
myform.data3ch1.Text

' goes til here
If szVoltageBinary = "0000000000000000" Then
myform.lblVoltageCh1 = "0000.00 mV"
myform.lblTempCh1 = "0000.00 °C"
Else
If dblCold > 0 Then
szVoltageDec = (dec24(szVoltageBinary) * 80000 / 16777215) + (42 *
dblCold)
' voltage + temperature find
buffer1 = CDBl(szVoltageDec)
myform.lblVoltageCh1.Caption = Mid(buffer1, 1, 8) & " uV"
buffer2 = (buffer1 / 41)
myform.lblTempCh1.Caption = Mid(buffer2, 1, 4) & " °C"
Else
MsgBox ("Please Run 'cold 100'")
End If
End If
End If
End If
End Function
Function Cactus(databits As String) As String
Dim first, second, third As String
Dim buffer As String
Dim bufferdec As Double
If Len(databits) = 3 Then
first = bin(Asc(Mid(databits, 1, 1)))
second = bin(Asc(Mid(databits, 2, 1)))
third = bin(Asc(Mid(databits, 3, 1)))
buffer = first & second & third
bufferdec = dec24(buffer)
bufferdec = bufferdec * 80000 / 16777215 / 41 + dblCold
bufferdec = bufferdec
Cactus = bufferdec
End If
End Function

------------------------EOF------------------------

MS-DOS Name: SortData.cls VB Name: SortData

Dim count As Long 'global
Public Sub Filter(ByVal szString As String)
Dim st, ft, md, of, gn As Integer
st = InStr(1, szString, "st", vbTextCompare) + 2
ft = InStr(1, szString, "ft", vbTextCompare) + 2

155
md = InStr(1, szString, "md", vbTextCompare) + 2
gn = InStr(1, szString, "gn", vbTextCompare) + 2
of = InStr(1, szString, "of", vbTextCompare) + 2

Registers(0) = bin(Asc(Mid(szString, st, 1)))
Registers(1) = bin(Asc(Mid(szString, md, 1)))
Registers(2) = bin(Asc(Mid(szString, md + 1, 1)))
Registers(3) = bin(Asc(Mid(szString, ft, 1)))
Registers(4) = bin(Asc(Mid(szString, ft + 1, 1)))

Registers(5) = bin(Asc(Mid(szString, gn, 1)))
Registers(6) = bin(Asc(Mid(szString, gn + 1, 1)))
Registers(7) = bin(Asc(Mid(szString, gn + 2, 1)))
Registers(8) = bin(Asc(Mid(szString, of, 1)))
Registers(9) = bin(Asc(Mid(szString, of + 1, 1)))
 Registers(10) = bin(Asc(Mid(szString, of + 2, 1)))
SetTheRegs
End Sub

Public Function Validate(ByVal szString As String) As Boolean
Dim st, ft, of, gn, md, good As Integer

If InStr(1, szString, "st", vbTextCompare) > 0 Then
good = 1
Else
Validate = False
Exit Function
End If

If InStr(1, szString, "gn", vbTextCompare) > 0 Then
good = 1
Else
Validate = False
Exit Function
End If

If InStr(1, szString, "md", vbTextCompare) > 0 Then
good = 1
Else
Validate = False
Exit Function
End If

If InStr(1, szString, "of", vbTextCompare) > 0 Then
good = 1
Else
Validate = False
Exit Function
End If

If InStr(1, szString, "ft", vbTextCompare) > 0 Then
good = 1
Else
Validate = False
Exit Function
End If

If good = 1 Then Validate = True
End Function
APPENDIX G: Calibration Electrical Diagram
APPENDIX H: System Hardware Snapshots
SELECTED BIBLIOGRAPHY


VITA AUCTORIS

Adrian Chepetan was born in 1954 in Arad, Romania. He received his Bachelor of Engineering from the Polytechnic Institute “Aurel Vlaicu”, Timisoara, Romania, in 1984. He is currently a candidate for the Master’s Degree of Applied Science in Electrical Engineering at the University of Windsor, Windsor, Ontario, Canada.