Project #: E-25-662
Center #: R6646-0A0
Contract#: AX-853025
Prime #: DE-AC09-76R00001

Active
Rev #: 0
OCA file #:
Work type : RES
Document : SUBCONT
Contract entity: GTRC

Subprojects ? : N
Main project #:

Project unit: ME
Project director(s): SCHNEIDER A

Unit code: 02.010.126
(404)894-3725

Sponsor/division names: SAVANNAH RIVER LABORATORY / E I DUPONT DE NEMOURS CO
Sponsor/division codes: 240 / 004

Award period: 881129 to 891201 (performance) 891201 (reports)

Sponsor amount
Contract value 142,619.00
Funded 142,619.00

Cost sharing amount 0.00

Does subcontracting plan apply ?: N

Title: REMOTE MONITORING OF RADIOACTIVE GLASS PRODUCT

PROJECT ADMINISTRATION DATA

OCA contact: Steven K. Watt 894-4820
Sponsor technical contact
R. SCHUMACHER
(803)725-4184
E.I. DUPONT DE NEMOURS & COMPANY
SAVANNAH RIVER PLANT
AIKEN, SC 29808-0001

M.B. SPLETZER
(803)725-3568
E.I. DUPONT DE NEMOURS & COMPANY
SAVANNAH RIVER PLANT
AIKEN, SC 29808-0001

GA TECH RETAINS TITLE TO PROPERTY COSTING < $1K WITH PRIOR APPROVAL FROM SPON

Security class (U,C,S,TS) : U
Defense priority rating : N/A
Equipment title vests with: Sponsor X GIT

ONR resident rep. is ACO (Y/N): Y
N/A supplemental sheet

Administrative comments -
PROJECT INITIATION.
NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 07/19/90

Project No. E-25-662
Center No. R6646-0AO

Project Director SCHNEIDER A
School/Lab MECH ENGR

Sponsor SAVANNAH RIVER COMPANY/WESTINGHOUSE

Contract/Grant No. AX-853025
Contract Entity GTRC

Prime Contract No. DE-AC09-76SR00001

Title REMOTE MONITORING OF RADIOACTIVE GLASS PRODUCT

Effective Completion Date 900301 (Performance) 900301 (Reports)

Closeout Actions Required:

<table>
<thead>
<tr>
<th>Action</th>
<th>Y/N</th>
<th>Date Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Invoice or Copy of Final Invoice</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Final Report of Inventions and/or Subcontracts</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Government Property Inventory &amp; Related Certificate</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Classified Material Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Release and Assignment</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

Comments

Subproject Under Main Project No.

Continues Project No.

Distribution Required:

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Director</td>
<td>Y</td>
</tr>
<tr>
<td>Administrative Network Representative</td>
<td>Y</td>
</tr>
<tr>
<td>GTRI Accounting/Grants and Contracts</td>
<td>Y</td>
</tr>
<tr>
<td>Procurement/Supply Services</td>
<td>Y</td>
</tr>
<tr>
<td>Research Property Management</td>
<td>Y</td>
</tr>
<tr>
<td>Research Security Services</td>
<td>N</td>
</tr>
<tr>
<td>Reports Coordinator (OCA)</td>
<td>Y</td>
</tr>
<tr>
<td>GTRC</td>
<td>Y</td>
</tr>
<tr>
<td>Project File</td>
<td>N</td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
</tr>
</tbody>
</table>

NOTE: Final Patent Questionnaire sent to PDPI.
REMOTE MONITORING OF RADIOACTIVE GLASS PRODUCT

PHASE 1
PROGRESS REPORT

PROJECT E25-662
Period Covered: November 29, 1988 - June 1, 1989

Project Director: Alfred Schneider

August 1989

Work done under Subcontract No.AX-853025 with Westinghouse Savannah River Company
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. The Remote Monitoring Concept</td>
<td>5</td>
</tr>
<tr>
<td>3. The Measurement of Viscosity</td>
<td>7</td>
</tr>
<tr>
<td>3.1 General</td>
<td>7</td>
</tr>
<tr>
<td>3.2 Cannon-Fenske Viscometry</td>
<td>8</td>
</tr>
<tr>
<td>3.3 Viscometry by Bubble Rise Velocity</td>
<td>10</td>
</tr>
<tr>
<td>4. The Generation of Helium Bubbles</td>
<td>11</td>
</tr>
<tr>
<td>4.1 General Requirements</td>
<td>11</td>
</tr>
<tr>
<td>4.2 Timed Flow Control Method</td>
<td>13</td>
</tr>
<tr>
<td>4.3 Metered Mass Method</td>
<td>15</td>
</tr>
<tr>
<td>5. The Determination of the Specific Gravity of Liquids</td>
<td>19</td>
</tr>
<tr>
<td>6. Bubble Measurements and Characterization</td>
<td>19</td>
</tr>
<tr>
<td>7. Adaptation to Remote Operation</td>
<td>21</td>
</tr>
<tr>
<td>8. Computer Control, Data Acquisition and Processing</td>
<td>26</td>
</tr>
<tr>
<td>9. Future Work</td>
<td>28</td>
</tr>
<tr>
<td>10. Conclusions</td>
<td>29</td>
</tr>
</tbody>
</table>
PROJECT PERSONNEL
(All Part-Time)

Alfred Schneider, Ph.D.  
Project Director and Professor

Michael Hayes, Ph.D.  
Senior Research Scientist

Robert Abraham, B.A.  
Laboratory Supervisor

K. K. Li, M.N.E.  
Graduate Research Assistant

K. Choi, M.Ch.E.  
Graduate Research Assistant
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Project Objectives</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Constant Property Lines for the Na$_2$O - B$_2$O$_3$ - SiO$_2$ System at 1100°C</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Viscosity of Pure Glycerol</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>Viscosity of S-2000 Oil</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>Comparison of Reduced Reciprocal BRVs and Reduced (CF) Viscosities</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>Bubble Generation by Timed Flow Control Method</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>Bubble Generation by Metered Mass Method</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>Relation between Bubble Diameter and Upstream He Pressure</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>Original and Modified Bubble Launchers</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>Helium Bubbles in S-2000 Oil</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>Remote Glass Monitoring Concept</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td>Carrier Gas Delivery by Suction Method</td>
<td>24</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

The conversion of High-Level Liquid Wastes (HLLW) arising from the reprocessing of spent nuclear fuels into solid forms is necessary to prevent the leakage of stored radioactive materials into the soil and to prepare the long-lived radioactive wastes for permanent disposal in geologic formations. Glass has been the preferred waste form because of its relative chemical stability, capability of incorporating various elements and compounds, and extensive experience with its production. Vitrification facilities for the conversion of HLLW, primarily into borosilicate glass, have been in operation for several years and additional ones are in an advanced state of construction in many countries.

The properties of a glass are determined primarily by its composition, operating variables during production, and the cooling rate after the glass is poured into the canisters. The chemical stability of a glass containing radioactive ingredients, as evidenced by a low leaching rate for most elements, is the property of paramount concern. Since variations in the chemical composition of the glass product may cause unacceptable variations in the leaching rates of certain radionuclides, great care is taken during the design of vitrification facilities to allow for very precise compounding of the ingredients and for continuous monitoring of sensitive operating variables. The high quality control standards in the production of the glass are part of the overall qualification process which also includes the canisters, transportation casks, and geologic host repository.

It became clear several years ago that the regulatory authorities will require that confirmatory test data be supplied for the glass shipped to a repository. At present, the only way in which such requirements could be met is to periodically remove glass samples which are chemically analyzed in a remotely operated analytical facility. Obtaining such samples is somewhat cumbersome because of the
complexity of the remotely operated sampling equipment. The facilities for the transport and handling of the highly radioactive samples are equally complex and may require frequent maintenance. It is also possible that, because of the delays in obtaining the results of sample analyses, some canisters may be filled with glass which is outside the established specification and the disposal of such off-spec material could pose a problem. Sampling can only be done at finite intervals and there would remain some uncertainty about the quality of the glass produced between samplings.

Great strides have been made in recent years, particularly in the chemical industry, in the development of on-line control methods which are valuable, not only in controlling various processes, but also in providing for the continuous monitoring of the quality of the product. The extension of such control methods to the production of radioactive waste glasses requires the availability of instruments which would continuously provide an analysis of the glass product about to be poured into the canister. This is a formidable problem, considering the very high temperature of the molten glass, the intense radioactivity, and the complex composition of the glasses.

A method which may provide a near-continuous indication of the composition range as well as deviations from a target product composition was proposed in 1986 by Alfred Schneider of the Georgia Institute of Technology. This concept is based on the ability to relate two or more physical properties of the molten glass with its composition. By reference to an experimentally developed data base, the determination of the physical properties of a specific type of glass makes possible the estimation of the composition of the glass. Most waste glass forms can be treated as ternary compounds and a specific composition point would then correspond to the intersection of two characteristic lines representing constant
values of two physical properties. Subsequent investigations showed that viscosity and specific gravity or viscosity and electrical resistivity would be suitable physical properties for this purpose. A conceptual design for a glass melter monitoring system incorporating the above ideas was developed in 1987 at Georgia Tech, as part of a Nuclear Engineering student design project.

An experimental study to demonstrate the practicality of the remote glass monitoring concept and to assess its applicability to the Defense Waste Production Facility under construction at the Savannah River Plant was initiated at Georgia Tech in December 1988 under the sponsorship of the E. I. DuPont Co., the SRP contractor at that time, and is continuing, at present, under the new contractor, the Westinghouse Savannah River Co.

The study has been divided into two phases, each scheduled for a period of six months. The objectives for the first phase included the development of equipment and techniques suitable for the remote determination of the viscosity and the specific gravity of molten glass, and of computer algorithms illustrating the estimation of glass compositions from the measurement of viscosity, specific gravity, and temperature. The methods and equipment development was to be done at moderate temperatures using liquids with properties similar to those of molten glass. In the second phase, the techniques developed in the first phase were to be validated for molten glass, including simulated SRP waste glasses.

The objectives for Phase 1 of this Project have been fully achieved, as can be seen from the details provided in this Progress Report. A schematic outline of the Study is shown in Figure 1.
Figure 1: Project Objectives
2. THE REMOTE MONITORING CONCEPT

Triangular phase diagrams are frequently used to represent ternary systems or systems which can be simplified to pseudo-ternary. Properties of borosilicate glasses can be shown on a ternary diagram in which SiO₂, Na₂O, and B₂O₃ are the constituent components. The SRP waste glass can also be represented on a triangular phase diagram on which PHA (hydrolysis product from the Cs removal process), glass frit (having a specific composition), and sludge are the primary constituents. Similarly, the West Valley glass can be treated as a pseudo-ternary system with zeolite, sludge, and glass formers as the constituents.

Variation of physical properties with composition can be represented on a triangular diagram as constant value lines for a given temperature as shown in Figure 2. By determining two physical properties at the same temperature, one can derive the composition by locating the point where the two lines of constant property intersect. The location of the point of intersection is facilitated if the respective lines representing the two properties do not intersect at small angles.

The results of a literature search for three physical properties of borosilicate glasses are shown in Figure 2. It can be seen that melt viscosity varies appreciably with composition, while the specific gravity changes only slightly. The electrical resistivity is also quite composition-dependent, but to a lesser extent than viscosity. It is also apparent that the viscosity/specific gravity and the viscosity/electrical resistivity pairs are quite suitable, while the specific gravity/electrical resistivity combination would produce larger uncertainties in the derived composition values. It was decided to concentrate on the viscosity/specific gravity pair during this phase of the project.
Fig. 2. Constant property lines for the Na$_2$O - B$_2$O$_3$ - SiO$_2$ system at 1100°C
The selection of the properties to be investigated was also influenced by the availability of measuring techniques amenable to remote operation in a high radiation field and at elevated temperature. While suitable methods are available for the determination of specific gravity and electrical resistivity under these conditions, no practical methods for the measurement of viscosity were known which could be adapted for this application. A novel method, based on the dependence on viscosity of the rise velocity of a gas bubble in a liquid, was conceived and experimentally demonstrated.

In its present version, the remote glass monitoring concept consists of a mechanism for introducing carefully sized helium bubbles into the molten glass, the remote measurement of the bubble rise velocity, and the simultaneous determinations of the specific gravity and temperature of the glass in the immediate vicinity of the bubble rise path. Data processing with a computer includes the calculation of temperature corrections and the estimation of glass composition, using an experimentally obtained data base and suitable algorithms. If further developments are successful, measurements could be made at one to two minutes intervals. The eventual probe is envisioned to consist of Inconel tubing in an envelope of 0.5 to 1 inches in diameter.

3. THE MEASUREMENT OF VISCOSITY

3.1 General

Many methods are available for the measurement of the viscosity of liquids. The high temperature at which measurements are carried out with molten glass and the corrosiveness of the liquid have restricted the choice to the falling ball, flow rate through an orifice, and the rotating spindle methods. The latter method, generally consisting of a Brookfield type instrument adapted for high-temperature use, is the
method favored for the determination of the viscosity of molten glass. It became quite obvious at the outset of this study, that a Brookfield type viscometer could not be adapted for the continuous monitoring of glass melts, especially for remote operation in an intense radioactive field. Other methods, such as the falling ball method or the measurement of flow rate through an orifice, would also be difficult to develop for the required application. Attention was, therefore, focussed on a novel measurement method based on the rise velocity of a helium bubble in the liquid.

3.2 Cannon-Fenske Viscometry

Evaluation of the suitability of the bubble rise velocity viscometry required a comparison with viscosity data obtained using a standard method.

Two liquids were used during the first phase of this study to simulate the molten glass: CP grade glycerol and Cannon S-2000 Standard oil.

A data base was developed for the viscosities of these liquids, using a set of certified Cannon-Fenske viscometers. The viscosities of both liquids were determined at temperatures ranging from 10 to 60°C and covering a viscosity range between 291 and 15,075 centipoise. The data obtained generally agreed to within 5% with the corresponding data in the literature or with the supplier's certifications. Work with glycerol was discontinued when it was found that its hygroscopicity led to rapid absorption of water which sharply affected the viscosity. The kinematic viscosity (in centistokes) as determined with the Cannon-Fenske viscometers is converted to the dynamic viscosity (in centipoise), by multiplying the former by the corresponding specific gravity.

Results obtained for glycerol and S-2000 oil are shown in Figures 3 and 4.
Figure 3: Viscosity of Pure Glycerol

Figure 4: Viscosity of S-2000 Oil
3.3 **Viscometry By Bubble Rise Velocity**

The velocity of gas bubbles rising in a liquid has been shown to be a function of the densities and viscosities of the gas and the liquid, the diameter of the gas bubble, and the gravitational constant. Other factors, such as a change in size of the ascending bubble due to diffusion, chemical reaction, or expansion caused by changes in temperature and hydrostatic pressure, may also have an effect which can be minimized by proper selection of the gas and of the range over which measurements are made.

Quantitative relations for small bubbles were shown to agree with the Stokes or with the Hadamard-Rybczynski formula which are similar, except for a constant which is 1/3 in the former and 2/9 in the latter formula.

Helium bubble rise velocities (BRV) were determined at several temperatures for S-2000 oil, after ascertaining that the bubble diameters can be properly controlled and reproduced. The results are shown below:

<table>
<thead>
<tr>
<th>TEMPERATURE</th>
<th>BRV VISCOSITY</th>
<th>CF VISCOSITY</th>
<th>DIFFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>°C</td>
<td>cm/sec</td>
<td>poise</td>
<td>poise</td>
</tr>
<tr>
<td>22.5</td>
<td>1.14</td>
<td>70.83</td>
<td>68.00</td>
</tr>
<tr>
<td>26.0</td>
<td>1.78</td>
<td>45.01</td>
<td>49.50</td>
</tr>
<tr>
<td>33.0</td>
<td>3.22</td>
<td>24.90</td>
<td>26.80</td>
</tr>
<tr>
<td>40.0</td>
<td>6.00</td>
<td>13.23</td>
<td>15.80</td>
</tr>
</tbody>
</table>

*Calculated by the Hadamard-Rybczynski formula*
Dynamic viscosities calculated by the Hadamard-Rybczynski formula are generally lower than the corresponding viscosities obtained with the Cannon-Fenske viscometer (CF). This suggests that, as more experimental data are obtained, an empirically derived relation will be slightly different than either of the previously cited formulae. Because of this uncertainty, a comparison of the BRV and CF viscosity measurements was made by calculating reduced reciprocal velocities (= velocity at 25° divided by the velocity at a given temperature) and reduced CF dynamic viscosities (= viscosity at a given temperature divided by the viscosity at 25°). The results are shown below and also in Figure 5.

<table>
<thead>
<tr>
<th>TEMPERATURE °C</th>
<th>BRV cm/sec</th>
<th>(BRV)_{25}/(BRV)_t</th>
<th>( \eta_t/\eta_{25} ) (CF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.5</td>
<td>1.14</td>
<td>1.40</td>
<td>1.25</td>
</tr>
<tr>
<td>25.0</td>
<td>1.60</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>26.0</td>
<td>1.78</td>
<td>0.90</td>
<td>0.90</td>
</tr>
<tr>
<td>33.0</td>
<td>3.22</td>
<td>0.50</td>
<td>0.49</td>
</tr>
<tr>
<td>40.0</td>
<td>6.0</td>
<td>0.27</td>
<td>0.28</td>
</tr>
</tbody>
</table>

The two reduced values compare quite favorably at all but one temperature.

4. THE GENERATION OF HELIUM BUBBLES

4.1 General Requirements

For the purpose of measuring the viscosity of liquids by the bubble rise velocity method, it is necessary that the system for generating the gas bubbles meet the following requirements:

The mass of gas delivered must be controllable and the variability should be as low as possible.
Figure 5: Comparison of Reduced Reciprocal BRVs and Reduced (CF) Viscosities
The exact instance of bubble launch must be known.

The bubble size shall be determined by the mass of gas delivered and by the hydrostatic pressure at the point of launch and be unaffected by intrinsic properties of the liquid, such as viscosity and surface tension.

The bubble diameter should be such that the rise velocity is sufficient to complete a measurement cycle in less than one minute.

All components of the generating system which require adjustment or which are subject to radiation damage must be located at some distance from the liquid container.

The gas should be inert, have a low solubility in the liquid, and be amenable to a remote method for determining the rise velocity.

Helium was the gas selected, because of its chemical inertness and the availability of detection equipment for helium which could be adapted to remote operation.

Two methods were investigated for the delivery of precisely measured quantities of helium: a timed flow control and a metered mass method. The latter method was adopted because of its simplicity, excellent reproducibility, adjustability of bubble diameter, and adaptability to computer control and remote operation.

4.2 Timed Flow Control Method

The timed flow control method utilizes a precision gas mass flow controller and timer to deliver a desired mass of helium through the launch tube into the liquid. A schematic diagram of this system is shown in Figure 6.
Figure 6: Bubble Generation by Timed Flow Control Method

Figure 7: Bubble Generation by Metered Mass Method
The desired flow rate is obtained by adjusting the voltage applied to the controller. An adjustable but constant pressure of helium was maintained upstream of the flow controller. Gas flow was initiated and terminated by signals from the computer at preset time intervals.

Difficulties were encountered because of the inability to achieve instantaneously the preset flow rate, a requirement if the flow period was to be kept very brief to obtain a sharp flow pulse. It also appeared that the helium delivered would not always emerge as a single bubble, which made control of bubble size difficult. The problem with this method may have been further aggravated by the choice of a very small diameter (0.038") delivery tube, in an attempt to minimize holdup and thus sharpen the flow pulse. However, the greater resistance to flow yielded the opposite effect. When flow rate, He pressure, and time interval were properly adjusted, bubbles were delivered which had a variability in rise velocity (= standard deviation for 20 replicate measurements) ranging from 6.9 to 12.1% at various liquid temperatures.

4.3 Metered Mass Method

The apparatus for the metered mass method of generating He bubbles consists of a small chamber (1.15cc) enclosed by two fast-acting solenoid valves, as shown schematically in Figure 7. He is introduced into the chamber at pressures ranging from 10 to 25 psig by opening the upstream valve while the downstream valve remains closed. The upstream valve is closed and the downstream valve is opened to deliver a mass of gas determined by the difference between the upstream and downstream pressures. If liquid is present in the tube, it is gradually expelled by successive pulses; the downstream pressure increases, until gas bubbles are actually released into the liquid. Subsequently, every cycle produces a bubble which is
released almost instantaneously at the opening of the downstream valve. The rapid pulse is brought about by the much higher pressure of the gas in the metering chamber, and the momentum imparted by the rapid expansion of the gas causes the bubble to be ejected like a projectile. Under these circumstances, the properties of the liquid (viscosity and surface tension) have a much smaller effect. The variability of bubble rise velocities at different temperature ranged from 1.4 to 3.2%.

Bubble diameters, as estimated from photographs, ranged from 0.76 to 1.65cm. As expected, the diameter was found to be a linear function of the cube root of the upstream and downstream pressures. The results for one series of measurements with S-2000 oil at 25\degree and a bubble release point 10.7cm below the surface are shown in Figure 8. Correlation of the data by the least square method gave the following equation:

\[
D_B = 0.5139 \sqrt[3]{P} - 0.133 + 0.1689
\]

where \(D_B\) = bubble diameter, cm

\(P\) = upstream He pressure, psig

It was found that the shape of the launch tube is very important. Tubes which are turned upwards (U-shaped, 180\degree) were found to contain residual liquid which periodically resulted in a malformed bubble caused by entrapped liquid. This situation was remedied by using a tube with a 90\degree bend which gives a slightly curved initial bubble trajectory but which prevents the trapping of liquid in the tube. The modification is shown schematically in Figure 9. Optimization of the launching tubes will be studied during the next phase of this project.
Figure 8: Relation between Bubble Diameter and Upstream He Pressure
Figure 9: Original (a) and Modified (b) Bubble Launchers

trapped liquid
5. **THE DETERMINATION OF THE SPECIFIC GRAVITY OF LIQUIDS**

The widely used method based on measuring the difference in hydrostatic pressures at two levels can be combined with the system consisting of two bubble launching tubes described in Section 7. Transducers which have been integrated with the bubble generating system were used for determining the specific gravity of S-2000 oil at temperatures ranging from 5 to 50°C. A value of 0.90 g/cc was determined at 25°C, which compares with the value of 0.8732 given in the supplier's data sheet. Further refinement of the measurement methods are planned for the second phase of this study.

6. **BUBBLE MEASUREMENTS AND CHARACTERIZATIONS**

During the development of a system for the generation of He bubbles, it became necessary to have a technique for assessing the degree to which bubble sizes could be controlled. From the equations which have been found to represent satisfactorily the rise velocity of gas bubbles in liquids, it was evident that if the viscosity and specific gravities of the liquid are kept constant, which would be the case if the temperature is carefully controlled, the rise velocity is proportional to the square of the bubble diameter. Thus, the degree of size uniformity can be determined from the distribution of rise velocities in the same liquid at constant temperature.

Bubble shapes and diameters were determined photographically with a Nikon camera. Typical bubbles are shown in Figure 10. In general, the bubbles were approximately spherical in shape, with a pear shape being noticed immediately after launch.
Figure 10: Helium Bubbles in S-2000 Oil.
All rise velocity measurements were made visually, by timing the interval between the rise of the bubble past two index lines. This method was not very precise, since at lower viscosities the time intervals were on the order of only one second. With the development of instrumental measurement methods, the precision is expected to improve by eliminating subjective operator errors. Multiple exposure photography, using the precision timer of the Nikon camera, have also been used to estimate rise velocity, as shown in Figure 10.

7. ADAPTATION TO REMOTE OPERATION

The success of the concept proposed for the monitoring of waste glass composition is linked with the ability to adapt the near continuous determination of glass viscosity and specific gravity by remote techniques. As was shown above, the bubble generating system and the determination of the specific gravity of the liquid are readily adaptable to remote operation. The major challenge of this study was found to be the development of a remote method for the measurement of the bubble rise velocity.

The approach pursued consists of determining the exact arrival time of a helium bubble at the liquid-gas interface using a He detector. Several types of gas detectors are suitable for this purpose; the differential thermal conductivity cell and the mass spectrometer type leak detector are two widely used instruments. A Varian 936-40 portable He leak detector was used in this study. The equipment arrangement is shown schematically in Figure 11. The helium bubble emerging from the liquid is captured in a bonnet which is partially immersed in the liquid. A nitrogen carrier stream is introduced into the bonnet so as to promote the flushing of the helium into a transfer tube which is connected to the He detector
Figure 11: Remote Glass Monitoring Concept
located at a distance of several feet. The arrival of the helium pulse is signaled by the \textbf{He} detector as an increase in output voltage which is proportional to the He concentration in the nitrogen carrier. By carefully controlling the flow rate of the carrier gas, and thus the travel time of the He spike between bonnet and detector, it is possible to calculate the exact arrival time of the bubble at the liquid-gas interface. The rise velocity measurement is further simplified by employing two launch tubes which introduce He bubbles of equal mass but at different levels. Since the spacing of the launch positions is known, and the difference between the total time elapsed from the introduction to the detection of the two He bubbles is measured, the average rise velocity between the two levels can be calculated by dividing the time difference by the level difference. The bonnet location and carrier gas introduction were subsequently modified, to prevent depressing of the liquid by excessive pressurization. The bonnet bottom is kept out of the liquid and air, which is now the carrier gas, is transported to the detector by suction produced with a metering pump. The modified system is shown in Figure 12.
Figure 12: Carrier Gas Delivery by Suction Method
The calculational procedure with reference to Figure 11, is as follows:

\[ r = \text{average radius of bubbles 1 and 2 (controlled by bubble generators)} \]
\[ t_1 = \text{time for bubble 1 to traverse distance } L_1 \text{ (measured)} \]
\[ t_2 = \text{time for bubble 2 to traverse distance } L_2 \text{ (measured)} \]
\[ d = \text{difference between levels of bubble launch (known)} \]
\[ L_1 = \text{vertical distance of bubble 1 launch point from liquid-gas interface (varies with liquid level)} \]
\[ L_2 = \text{vertical distance of bubble 2 launch point from liquid-gas interface (varies with liquid level)} \]
\[ \Delta P = \text{differential hydrostatic pressure between bubble launch points (measured)} \]
\[ t = \text{time for bubble 2 to traverse distance } d \text{ (calculated)}. \]

**General Relation (simplified):**

\[ \eta = \frac{k \rho g r^2}{v} \]

where

\[ v = \text{average rise velocity of bubbles 1 and 2 (calculated)} \]
\[ \eta = \text{viscosity of liquid (calculated)} \]
\[ \rho = \text{specific gravity of liquid (calculated)} \]
\[ g = \text{gravitational constant (known)} \]
\[ k = \frac{2}{9} \text{ in Stoke's formula or } k = \frac{1}{3} \text{ in Hadamard-Rybczynski formula.} \]

But

\[ t = t_2 - t_1 \]
\[ v = \frac{d}{t_2 - t_1} \]
\[ \rho = \frac{\Delta P}{gd} \]

so

\[ \eta = \frac{k \left( \frac{\Delta P}{gd} \right) gr^2}{\left\{ \frac{d}{t_2 - t_1} \right\}} = \frac{k (\Delta P) r^2 (t_2 - t_1)}{d^2} \]

25
Computer control of bubble generation and detection is accomplished with an IBM-AT compatible computer together with Omega analog and digital I/O cards. One digital output channel is used to energize the relay which initiates the bubble launch by actuating the He control valves; analog input channels (one for each signal) are used to record the temperature of the liquid as measured with a thermocouple, the pressures in the bubble launch tubes obtained with pressure transducers, and the helium concentration in the carrier gas as indicated by the He detector output voltage. The control program written in BASIC was compiled with the QuickBasic 3.00 compiler.

A bubble generation and detection sequence proceeds as follows:

Immediately after starting the timer, a digital output channel is turned on energizing the SPDT relay which controls the He flow valves. When the relay is energized, the upstream valve closes and the downstream valve opens, ejecting a bubble into the liquid. After a brief time interval, the digital output channel is turned off, de-energizing the relay which causes the He flow valve to reset. The time at which the relay is de-energized is not important, since the bubble is ejected immediately after opening the downstream valve.

The arrival of the He in the carrier gas at the He detector is indicated by an increase in the detector output voltage which is transmitted to the computer. A 100-point running average is calculated to eliminate false signals due to noise spikes. When a rise in He level of 5% above the running average is detected, the timer is read and the total time elapsed between energizing the relay and detection of the He spike is recorded. The bubble rise time is then calculated by subtracting from the total time the previously determined delay due to bubble breakage and transport of the He from the bonnet to the detector.
The frequency at which bubbles can be launched into the liquid is determined by the total time elapsed between energizing the relay and detection of the He. With the system used in this phase of the study, one bubble can be launched about every twenty seconds.

The program TERNARY written in FORTRAN is used to estimate the composition of a ternary system from two physical properties. Experimentally determined property values at a given temperature are plotted on a ternary diagram (or on a simplified two-dimensional plot) and lines of constant property values are drawn on the diagram by interpolation between the experimental points. Polynomial equations are fitted to the lines by the least-square method. Specific compositions which correspond to the roots of the equations representing the pairs of property values are then obtained, using the Newton-Raphson Method. Details of the mathematical derivation and the computer program will be provided in the next Progress Report.

The program was tested with data obtained from the literature for sodium borosilicate glasses. Shown below are the compositions for which the viscosity and specific gravity data were obtained and the corresponding compositions estimated by the computer program:

<table>
<thead>
<tr>
<th>COMPOSITION TAKEN, mol%</th>
<th>COMPOSITION CALCULATED, mol%</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{B}_2\text{O}_3 )</td>
<td>( \text{SiO}_2 )</td>
</tr>
<tr>
<td>20.0</td>
<td>61.9</td>
</tr>
<tr>
<td>25.0</td>
<td>61.0</td>
</tr>
<tr>
<td>6.0</td>
<td>76.3</td>
</tr>
<tr>
<td>2.0</td>
<td>80.7</td>
</tr>
<tr>
<td>21.7</td>
<td>69.4</td>
</tr>
<tr>
<td>11.3</td>
<td>81.0</td>
</tr>
</tbody>
</table>
9. FUTURE WORK

The main objectives of the first phase of this study, the demonstration of the practicality of the remote monitoring concept and the development of computer programs for the estimation of the composition of ternary systems from two physical properties, have been successfully attained, using substitute liquids to simulate molten glass.

The plans for the second phase include the essential demonstration of the applicability of the methods, developed during the first phase, to simulated waste glasses. A high-temperature Lindberg furnace and associated precision temperature controller were obtained and tested. A new bubble launch system, incorporating Inconel bubble launching tubes of an improved design, is being fabricated and the carrier gas transport will be modified by the addition of a precision flow metering pump. An alternate bubble detection method, based on the change in electrical conductivity which occurs when a bubble passes through a conductivity cell placed in molten glass, will also be tested.

The computer program for the estimation of glass compositions and the computer control software will be further refined. Samples of simulated waste glass for which proper characterizations exist will be obtained from the Savannah River Laboratory and West Valley Nuclear Services. These samples will be tested with our high-temperature remote viscosity and specific gravity measuring system. The results will be compared with the properties furnished by the glass suppliers.
10. CONCLUSIONS

During the first phase of the project we have succeeded in developing a novel method for the measurement of viscosity which can be adapted to remote use with molten glass. The viscosity of a glass melt combined with the specific gravity at the same temperature can be used to estimate the composition of the glass. Computer programs for the control of the test system and for calculating the composition of a ternary compound have been developed and indications are that a level of precision for the estimated compositions can be achieved which will compare favorably with results obtainable for radioactive glasses by chemical analyses.
REMOTE MONITORING OF
RADIOACTIVE GLASS PRODUCT

FINAL REPORT

PROJECT E25-662

Period Covered: November 29, 1988 - December 1, 1989

PROJECT DIRECTOR: A. SCHNEIDER

June 1990

Work done under Subcontract No. AX-853025 with
Westinghouse Savannah River Company
PROJECT PERSONNEL
(All Part-Time)

Alfred Schneider, Ph.D.  Project Director and Professor
Michael Hayes, Ph.D.  Senior Research Scientist
Robert Abraham, B.A.  Laboratory Supervisor
K.K. Li, M.N.E.  Graduate Research Assistant
K.Choi, M.Ch.E.  Graduate Research Assistant
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. SELECTED PROPERTIES OF WASTE GLASSES</td>
<td>4</td>
</tr>
<tr>
<td>3. ESTIMATION OF GLASS COMPOSITION BY ANALYSIS OF PHYSICAL PROPERTIES</td>
<td>7</td>
</tr>
<tr>
<td>3.1 The Determination of the Density of Liquids</td>
<td>8</td>
</tr>
<tr>
<td>3.2 The Measurement of Viscosity</td>
<td>10</td>
</tr>
<tr>
<td>3.2.1 General</td>
<td>10</td>
</tr>
<tr>
<td>3.2.2 Cannon-Fenske Viscometry</td>
<td>10</td>
</tr>
<tr>
<td>3.2.3 Viscometry by Bubble Rise Velocity (BRV)</td>
<td>11</td>
</tr>
<tr>
<td>4. DEVELOPMENT OF THE BUBBLE RISE VELOCITY METHOD</td>
<td>16</td>
</tr>
<tr>
<td>4.1 General Requirements</td>
<td>16</td>
</tr>
<tr>
<td>4.2 Generation of Helium Bubbles</td>
<td>17</td>
</tr>
<tr>
<td>4.2.1 Timed Flow Control Method</td>
<td>17</td>
</tr>
<tr>
<td>4.2.2 Metered Mass Method</td>
<td>19</td>
</tr>
<tr>
<td>4.2.3 Bubble Characterization</td>
<td>23</td>
</tr>
<tr>
<td>4.3 Pressure Measurements</td>
<td>25</td>
</tr>
<tr>
<td>4.4 Measurement of Bubble Rise Velocity</td>
<td>30</td>
</tr>
<tr>
<td>5. EQUIPMENT</td>
<td>35</td>
</tr>
<tr>
<td>6. COMPUTER CONTROL, DATA ACQUISITION AND PROCESSING</td>
<td>41</td>
</tr>
<tr>
<td>6.1 Bubble Generation and Detection</td>
<td>42</td>
</tr>
<tr>
<td>6.2 Graphic Performance Display</td>
<td>44</td>
</tr>
<tr>
<td>6.3 Computer Software</td>
<td>44</td>
</tr>
<tr>
<td>6.3.1 General</td>
<td>44</td>
</tr>
<tr>
<td>6.3.2 A/D Card Driver</td>
<td>48</td>
</tr>
<tr>
<td>6.3.3 Low-Level Routines</td>
<td>49</td>
</tr>
<tr>
<td>6.3.4 High-Level Modules</td>
<td>51</td>
</tr>
<tr>
<td>6.3.5 Graphics Package</td>
<td>51</td>
</tr>
<tr>
<td>6.4 Estimation of Glass Composition</td>
<td>52</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (CONT)

7. RESULTS 53

7.1 Cannon-Fenske Viscosity Measurements 53
7.2 Bubble Size Determinations 53

7.2.1 Photographic Method 53
7.2.2 Pycnometric Method 53
7.2.3 Pressure Measurement Methods 54
7.2.4 Integration of Helium Concentration Method 54

7.3 Bubble Rise Velocity 62

8. FUTURE WORK 69

9. CONCLUSIONS 72

APPENDIX A Computer Program TERNARY

APPENDIX B Experimental Control and Data Analysis Programs
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.</td>
<td>Project Objectives</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2.</td>
<td>Constant Property Lines for the Na$_2$O-B$_2$O$_3$-SiO$_2$ System at 1100°C</td>
<td>6</td>
</tr>
<tr>
<td>Figure 3.</td>
<td>Density of SRL Waste Glass</td>
<td>9</td>
</tr>
<tr>
<td>Figure 4.</td>
<td>Viscosity of Pure Glycerol (Cannon-Fenske Method)</td>
<td>12</td>
</tr>
<tr>
<td>Figure 5.</td>
<td>Viscosity of S-2000 Oil (Cannon-Fenske Method)</td>
<td>12</td>
</tr>
<tr>
<td>Figure 6.</td>
<td>Comparison of Reduced Reciprocal BRVs and Reduced (CF) Viscosities</td>
<td>15</td>
</tr>
<tr>
<td>Figure 7.</td>
<td>Bubble Generation by Timed Flow Control Method</td>
<td>18</td>
</tr>
<tr>
<td>Figure 8.</td>
<td>Bubble Generation by Metered Mass Method</td>
<td>18</td>
</tr>
<tr>
<td>Figure 9.</td>
<td>Relation Between Bubble Diameter and Supply Pressure</td>
<td>21</td>
</tr>
<tr>
<td>Figure 10.</td>
<td>Original (a) and Modified (b) Bubble Launchers</td>
<td>22</td>
</tr>
<tr>
<td>Figure 11.</td>
<td>Helium Bubbles in S-2000 Oil</td>
<td>24</td>
</tr>
<tr>
<td>Figure 12.</td>
<td>Calibration of Pressure Transducers</td>
<td>26</td>
</tr>
<tr>
<td>Figure 13.</td>
<td>Profile of Pressure in Launch Tube During Normal Launch</td>
<td>28</td>
</tr>
<tr>
<td>Figure 14.</td>
<td>Profile of Pressure in Launch Tube (a) and He Concentration at Detector (b) During Abnormal Launches</td>
<td>29</td>
</tr>
<tr>
<td>Figure 15.</td>
<td>Remote Glass Monitoring Concept</td>
<td>31</td>
</tr>
<tr>
<td>Figure 16.</td>
<td>Carrier Gas Delivery by Suction Method</td>
<td>34</td>
</tr>
<tr>
<td>Figure 17.</td>
<td>High-Temperature Furnace and Test Setup</td>
<td>37</td>
</tr>
<tr>
<td>Figure 18.</td>
<td>Layout of Penetrations in Cover of High-Temperature Furnace</td>
<td>38</td>
</tr>
<tr>
<td>Figure 19.</td>
<td>Temperature Profile Inside Crucible</td>
<td>39</td>
</tr>
<tr>
<td>Figure</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>20</td>
<td>Bubble Launch Cycle</td>
<td>43</td>
</tr>
<tr>
<td>21</td>
<td>Photographs of Computer Monitor</td>
<td>45</td>
</tr>
<tr>
<td>22</td>
<td>Graphic Performance Display</td>
<td>46</td>
</tr>
<tr>
<td>23</td>
<td>Determination of Bubble Volume (a) and Diameter (b) by Pycnometry</td>
<td>55</td>
</tr>
<tr>
<td>24</td>
<td>Launch Tube Pressure Differences as a Function of Supply Pressure for SRL Waste Glass</td>
<td>56</td>
</tr>
<tr>
<td>25</td>
<td>Launch Tube Pressure Differences as a Function of Supply Pressure for S-2000 Oil</td>
<td>57</td>
</tr>
<tr>
<td>26</td>
<td>Launch Tube Pressure Differences as a Function of Supply Pressure for S-8000 Oil</td>
<td>58</td>
</tr>
<tr>
<td>27</td>
<td>Determination of Bubble Size by He Signal Integration</td>
<td>60</td>
</tr>
<tr>
<td>28</td>
<td>Gas Traveling Times as a Function of Temperature</td>
<td>63</td>
</tr>
<tr>
<td>29</td>
<td>Gas Traveling Times as a Function of Supply Pressure</td>
<td>64</td>
</tr>
<tr>
<td>30</td>
<td>Bubble Rise Velocity as a Function of Temperature for SRL Glass (Samples 3T-1, 3B, 4T-1/5-28-875/5)</td>
<td>65</td>
</tr>
<tr>
<td>31</td>
<td>Bubble Rise Velocities as a Function of Supply Pressure for SRL Glass (Samples 3T-1, 3B, 4T-1/5-28-875/5)</td>
<td>66</td>
</tr>
<tr>
<td>32</td>
<td>Experimental and Calculated (H-R) BRVs for S-2000 Oil as a Function of Temperature</td>
<td>67</td>
</tr>
<tr>
<td>33</td>
<td>Comparison of Reduced Reciprocal BRVs and Reduced Viscosities for SRL Waste Glass</td>
<td>68</td>
</tr>
<tr>
<td>34</td>
<td>Bubble Detection by Electrical Signal</td>
<td>70</td>
</tr>
<tr>
<td>35</td>
<td>Concept for an Integrated BRV Probe</td>
<td>71</td>
</tr>
<tr>
<td>Table 1.</td>
<td>BRV and CF Viscosities of Cannon S-2000 Oil</td>
<td>14</td>
</tr>
<tr>
<td>Table 2.</td>
<td>Comparison of Reduced Reciprocal BRVs and Reduced (CF) Viscosities</td>
<td>14</td>
</tr>
<tr>
<td>Table 3.</td>
<td>Test of Computer Program TERNARY</td>
<td>52</td>
</tr>
<tr>
<td>Table 4.</td>
<td>Effect of Temperature of LTPD Values</td>
<td>61</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

The conversion of High-Level Liquid Wastes (HLLW) arising from the reprocessing of spent nuclear fuels into a solid form is necessary to prevent the leakage of stored radioactive materials into the soil and to prepare the long-lived radioactive wastes for permanent disposal in geologic formations. Glass has been the preferred waste form because of its relative chemical stability, capability of incorporating various elements and compounds, and extensive experience with its production. Vitrification facilities for the conversion of HLLW, primarily into borosilicate glass, have been in operation for several years and additional ones are in an advanced state of construction in many countries.

The properties of a glass are determined primarily by its composition, operating variables during production, and the cooling rate after the glass is poured into the canisters. The chemical stability of a glass containing radioactive ingredients, as evidenced by a low leaching rate for most elements, is the property of paramount concern. Since variations in the chemical composition of the glass product may cause unacceptable variations in the leaching rates of certain radionuclides, great care is taken during the design of vitrification facilities to allow for very precise compounding of the ingredients and for continuous monitoring of sensitive operating variables. The high quality control standards in the production of the glass are part of the overall qualification process which also includes the performance of the canisters, the transportation casks, and the geologic host repository.
It was reorganized several years ago that the regulatory authorities will require that confirmatory test data be supplied for the glass shipped to a repository. At present, the only way in which such a requirement could be met is to periodically remove glass samples which are then chemically analyzed in a remotely operated analytical facility. Obtaining such samples is somewhat cumbersome because of the complexity of the remotely operated sampling equipment. The facilities for the transport and handling of the highly radioactive samples are equally complex and may require frequent maintenance. It is also possible that, because of the delays in obtaining the results of sample analyses, some canisters may be filled with glass which is outside the established specifications and the disposal of such off-spec material could pose a problem. Sampling can only be done at finite intervals and there would always remain some uncertainty about the quality of the glass produced between samplings.

Great strides have been made in recent years, particularly in the chemical industry, in the development of on-line control methods which are valuable not only in controlling various processes, but also in providing for the continuous monitoring of the quality of the product. The extension of such control methods to the production of radioactive waste glasses requires the availability of instruments which would continuously provide an analysis of the glass product about to be poured into the canister. This is a formidable problem, considering the very high temperature of the molten glass, the intense radioactivity, and the complex composition of the glasses.

A method which may provide a near-continuous indication of the composition range as well as deviations from a target product composition was proposed in 1986 by Alfred Schneider of the Georgia Institute of Technology. This concept
is based on the ability to relate two or more physical properties of the molten glass with its composition. By reference to an experimentally developed data base, the determination of the physical properties of a specific type of glass makes possible the estimation of the composition of the glass. Most waste glass forms can be treated as ternary compounds and a specific composition point would then correspond to the intersection of two characteristic lines representing constant values of two physical properties (isopleths). A review of the properties of borosilicate glasses showed that viscosity and density or viscosity and electrical resistivity would be suitable pairs of physical properties for this purpose. A conceptual design for a glass melter monitoring system incorporating the above ideas was developed in 1987 at Georgia Tech, as part of a Nuclear Engineering student design project.

An experimental study to demonstrate the practicality of the remote glass monitoring concept and to assess its applicability to the Defense Waste Production Facility (DWPF) under construction at the Savannah River Plant (SRP) was initiated at Georgia Tech in December 1988 under the sponsorship of the E.I. DuPont Co., the SRP contractor at that time. The study was continued under the new SRP contractor, the Westinghouse Savannah River Co.

The project was divided into two phases, each scheduled for a period of six months. The objectives for the first phase included the development of equipment and techniques suitable for the remote determination of the viscosity and the specific gravity of molten glass, and of computer algorithms illustrating the estimation of glass compositions from the measurement of viscosity, density, and temperature. The methods and equipment development was to be done at
moderate temperatures using liquids with properties similar to those of molten glass. In the second phase, the techniques developed in the first phase were to be validated for molten glass, including simulated SRP waste glasses.

The objectives of this project were fully achieved, as can be seen from the details provided in this Final Report. A schematic outline of the Study is shown in Figure 1.

2. SELECTED PROPERTIES OF WASTE GLASSES

Triangular phase diagrams are frequently used to represent ternary systems or systems which can be simplified to pseudo-ternary. Properties of borosilicate glasses can be shown on a ternary diagram in which SiO₂, Na₂O, and B₂O₃ are the constituent components. The SRP waste glass can also be represented on a triangular phase diagram on which PHA (hydrolysis product from the cesium removal process), glass frit (having a specific composition), and sludge (precipitates formed during the neutralization of the HLLW) are the primary constituents. Similarly, the glasses developed by the West Valley Demonstration Project can be treated as a pseudo-ternary system, with zeolite, sludge, and glass formers as the constituents.

Variation of physical properties with composition can be represented on a triangular diagram as constant value lines for a given temperature (isopleths) as shown in Figure 2. By determining two physical properties at the same temperature, one can derive the composition by locating the point where the two isopleths intersect. The location of the point of intersection is facilitated if the respective lines representing the two properties do not intersect at small angles.
Figure 1. Project Objectives
Figure 2. Constant Property Lines for the Na$_2$O-B$_2$O$_3$-SiO$_2$ System at 1100°C
The results of a literature search for three physical properties of borosilicate glasses are shown in Figure 2. It can be seen that melt viscosity varies appreciably with composition, while the density changes only slightly. The electrical resistivity is also quite composition-dependent, but to a lesser extent than viscosity. It is also apparent that the viscosity/density and the viscosity/electrical resistivity pairs are quite suitable, while the density/electrical resistivity combination would produce larger uncertainties in the derived composition values. It was decided to concentrate on the viscosity/density pair during this phase of the project.

3. ESTIMATION OF GLASS COMPOSITION BY ANALYSIS OF PHYSICAL PROPERTIES

The selection of the physical properties of the glass had to take into consideration the availability of measuring methods suitable for use at temperatures above 1000°C, in an intense radiation field, and in contact with corrosive molten glass. In addition, the measuring techniques also had to be amenable to remote operational control at distances in excess of 15-20 feet. While suitable methods are available for the determination of density and electrical resistivity under these conditions, no practical methods for the measurement of viscosity were known which could be adapted for this application. A novel method, based on the dependence on viscosity of the rise velocity of a gas bubble in a liquid, was conceived and experimentally demonstrated.

In its present version, the remote glass monitoring concept consists of a mechanism for introducing carefully sized helium bubbles into the molten glass from two launchers at different depths in the melt, the remote measurement of the bubble rise velocity, and the simultaneous determinations of the density and
temperature of the glass in the immediate vicinity of the bubble rise path. Data processing with a computer includes all the calculations relevant to determining rise velocity, density, and temperature, as well as the graphical display of processes associated with the launch and detection of the bubbles. Presently, a complete measurement, involving the launch and detection of bubbles from both launchers, can be made in less than one minute. The eventual probe is envisioned to consist of Inconel tubing in an envelope of 0.5 to 1 inch in diameter.

3.1 The Determination Of The Density Of Liquids

A widely used method for the remote determination of the density of liquids is based on the difference in hydrostatic pressures at two levels. This method can be combined with the system consisting of two bubble launching tubes described in Section 4.4. Transducers which have been integrated with the bubble generating system were used for determining the density of S-2000 oil at temperatures ranging from 5 to 50°C. A value of 0.90g/cc was determined at 25°C, which compares with the value of 0.8732 given by the supplier.

The density of SRL glass was determined remotely for the temperature range 1038 to 1095°C. The results are shown in Figure 3.

The pressure difference measurements which served as the basis for the calculation of the density were found to be very precise. Typical standard deviations for 50-70 points were less than 0.03%.
Figure 3. Density of SRL Waste Glass
3.2 The Measurement Of Viscosity

3.2.1 General

Many methods are available for the measurement of the viscosity of liquids. The high temperatures at which measurements must be carried out with molten glass and the corrosiveness of the liquid have restricted the choices to the falling ball, flow rate through an orifice, and the rotating spindle methods. The latter method, generally consisting of a Brookfield type instrument adapted for high-temperature use, is the technique favored for the laboratory determination of the viscosity of molten glass samples. It became quite obvious at the outset of this study, that a Brookfield type viscometer could not be adapted for the continuous monitoring of glass melts, especially for remote operation in an intense radioactive field. Other methods, such as the falling ball method or the measurement of flow rate through an orifice, would also be difficult to develop for the required application. Attention was, therefore, focused on a novel measurement method based on the rise velocity of a gas bubble through the liquid.

3.2.2 Cannon-Fenske Viscometry

Evaluation of the suitability of the bubble rise velocity viscometry required a comparison with viscosity data obtained using a standard method. Two liquids were used during the first phase of this study to simulate the molten glass: CP grade glycerol and Cannon S-2000 Standard oil. A data base was developed for the viscosities of these liquids, using a set of certified Cannon-Fenske viscometers. The viscosities of both liquids were determined at temperatures ranging from 10 to 60°C and covering a viscosity range between 291 and 15,075
centipoise. The data obtained generally agreed to within 5% with the corresponding data in the literature or with the supplier's certifications. Work with glycerol was discontinued when it was found that its hygroscopicity led to rapid absorption of water which sharply affected the viscosity. The kinematic viscosity (in centistokes), as determined with the Cannon-Fenske viscometers, is converted to the dynamic viscosity (in centipoise), by multiplying the former by the corresponding density. Results obtained for glycerol and S-2000 oil are shown in Figures 4 and 5.

3.2.3 Viscometry by Bubble Rise Velocity (BRV)

The velocity of gas bubbles in a liquid has been shown to be a function of the densities and viscosities of the gas and the liquid, the diameter of the gas bubble, and the gravitational constant. Other factors, such as a change in size of the ascending bubble due to diffusion, chemical reaction, or expansion caused by changes in temperature and hydrostatic pressure, may also have an effect which can be minimized by proper selection of the gas and of the range over which measurements are made. Quantitative relations for small bubbles were shown to agree with the Stokes or with the Hadamard-Rybczynski formula which are similar, except for a constant which is 1/3 in the former and 2/9 in the latter formula. (See Section 4.4)

Helium bubble rise velocities were determined visually at several temperatures for S-2000 oil, after ascertaining that the bubble diameters can be properly controlled and reproduced. The results are shown in Table 1.
Figure 4. Viscosity of Pure Glycerol (Cannon-Fenske Method)

Figure 5. Viscosity of S-2000 Oil (Cannon-Fenske Method)
Dynamic viscosities based on bubble rise velocities (BRV) and calculated by the Hadamard-Rybczynski formula are generally lower than the corresponding viscosities obtained with the Cannon-Fenske viscometer (CF). This suggests that as more experimental data are obtained, an empirically derived relation will be slightly different than either of the previously cited formulae. Because of this uncertainty, a comparison of the BRV and CF viscosity measurements was made by calculating reduced reciprocal velocities (= velocity at 25°C divided by the velocity at a given temperature) and reduced CF dynamic viscosities (= viscosity at a given temperature divided by the viscosity at 25°C). The results are shown in Table 2 and also in Figure 6.
Table 1. BRV and CF Viscosities of Cannon S-2000 Oil

<table>
<thead>
<tr>
<th>Temp (°C)</th>
<th>BRV (cm/sec)</th>
<th>BRV Viscositya (poise)</th>
<th>CF Viscosity (poise)</th>
<th>Difference (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.5</td>
<td>1.14</td>
<td>70.83</td>
<td>68.00</td>
<td>+ 4.2</td>
</tr>
<tr>
<td>26.0</td>
<td>1.78</td>
<td>45.01</td>
<td>49.50</td>
<td>- 9.1</td>
</tr>
<tr>
<td>33.0</td>
<td>3.22</td>
<td>24.90</td>
<td>26.80</td>
<td>- 7.1</td>
</tr>
<tr>
<td>40.0</td>
<td>6.00</td>
<td>13.23</td>
<td>15.80</td>
<td>- 16.2</td>
</tr>
</tbody>
</table>

*a calculated by the Hadamard-Rybczynski formula

Table 2. Comparison of Reduced Reciprocal BRVs and Reduced (CF) Viscosities

<table>
<thead>
<tr>
<th>Temp (°C)</th>
<th>BRV (cm/sec)</th>
<th>(BRV)25/(BRV)t</th>
<th>(CF)(\eta_t/\eta_{25})</th>
<th>Difference %</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.5</td>
<td>1.14</td>
<td>1.40</td>
<td>1.25</td>
<td>+ 12</td>
</tr>
<tr>
<td>25.0</td>
<td>1.60</td>
<td>1.00</td>
<td>1.00</td>
<td>0</td>
</tr>
<tr>
<td>26.0</td>
<td>1.78</td>
<td>0.90</td>
<td>0.90</td>
<td>0</td>
</tr>
<tr>
<td>33.0</td>
<td>3.22</td>
<td>0.50</td>
<td>0.49</td>
<td>+ 2</td>
</tr>
<tr>
<td>40.0</td>
<td>6.00</td>
<td>0.27</td>
<td>0.28</td>
<td>- 4</td>
</tr>
</tbody>
</table>
Figure 6. Comparison of Reduced Reciprocal BRVs and Reduced (CF) Viscosities
4. DEVELOPMENT OF THE BUBBLE RISE VELOCITY METHOD

4.1 General Requirements

For the purpose of measuring the viscosity of liquids by the bubble rise velocity method, it is necessary that the system for generating the gas bubbles meet the following requirements:

- The mass of gas delivered must be controllable and the variability should be as low as possible.

- The exact instance of bubble launch must be known.

- The bubble size shall be determined by the mass of gas delivered and by the hydrostatic pressure at the point of launch and be unaffected by intrinsic properties of the liquid, such as viscosity and surface tension.

- The bubble diameter should be such that the rise velocity is sufficient to complete a measurement cycle in less than one minute.

- All components of the generating system which require adjustment or which are subject to radiation damage must be located at some distance from the liquid being tested.

- The gas should be inert, have a low solubility in the liquid, and be amenable to a remote method for determining the rise velocity.
4.2 Generation Of Helium Bubbles

Helium was the gas selected, because of its chemical inertness and the availability of detection equipment for helium which could be adapted to remote operation.

Two methods were investigated for the delivery of precisely measured quantities of helium: a timed flow control and a metered mass method. The latter method was adopted because of its simplicity, excellent reproducibility, adjustability of bubble diameter, and adaptability to computer control and remote operation.

4.2.1 Timed Flow Control Method

The timed flow control method utilizes a precision gas mass flow controller and timer to deliver a desired mass of helium through the launch tube into the liquid. A schematic diagram of this system is shown in Figure 7.

The desired flow rate is obtained by adjusting the voltage applied to the controller. An adjustable but constant pressure of helium was maintained upstream of the flow controller. Gas flow was initiated and terminated by signals from the computer at preset time intervals.

Difficulties were encountered because of the inability to achieve instantaneously the preset flow rate, a requirement if the flow period was to be kept very brief to obtain a sharp flow pulse. It also appeared that the helium delivered would not always emerge as a single bubble, which made control of bubble size difficult. The problem with this method may have been further aggravated by the choice
Figure 7. Bubble Generation by Timed Flow Control Method

Figure 8. Bubble Generation by Metered Mass Method
of a very small diameter (0.038") delivery tube, in an attempt to minimize holdup and thus sharpen the flow pulse. However, the greater resistance to flow yielded the opposite effect. When flow rate, helium pressure, and time interval were properly adjusted, bubbles were delivered which had a variability in rise velocity (= standard deviation for 20 replicate measurements) ranging from 6.9 to 12.1% at various liquid temperatures.

4.2.2 Metered Mass Method

The apparatus for the metered mass method of generating helium bubbles consists of a small chamber (1.15cc) enclosed by two fast-acting solenoid valves, as shown schematically in Figure 8. Helium is introduced into the chamber at pressures ranging from 8 to 15 psig by opening the upstream valve, while the downstream valve, remains closed. The upstream valve is then closed and the downstream valve is opened to deliver a mass of gas determined by the difference between the upstream and downstream pressures. If liquid is present in the tube, it is gradually expelled by successive pulses; the downstream pressure increases, until gas bubbles are actually released into the liquid. Subsequently, every cycle produces a bubble which is released a brief instance after the opening of the downstream valve. The rapid pulse is brought about by the much higher pressure of the gas in the metering chamber, and the momentum imparted by the rapid expansion of the gas causes the bubble to be ejected like a projectile. Under these circumstances, the properties of the liquid (viscosity and surface tension) have a much smaller effect on the bubble launch mechanism. The variability of bubble rise velocities at different temperatures ranged from 1.4 to 3.2%.
Bubble diameters, as estimated from the photographs, ranged from 0.76 to 1.65 cm. As expected, the diameter was found to be a linear function of the cube root of the difference of upstream and downstream pressures. The results for one series of measurements with S-2000 oil at 25°C and a bubble release point 10.7 cm below the surface are shown in Figure 9. Correlation of the data by the least square method gave the following equation:

\[ D_B = \sqrt[3]{P - 0.133} + 0.1689 \]

where \( D_B \) = bubble diameter, cm

\( P \) = upstream He pressure, psig

It was found that the shape of the launch tube is very important. Tubes which are turned upwards (U-shaped, 180°) were found to contain residual liquid which periodically resulted in a malformed bubble caused by entrapped liquid. This situation was remedied by using a tube with a 90° bend which gives a slightly curved initial bubble trajectory but which prevents the trapping of liquid in the tube. The modification is shown schematically in Figure 10. Considerable time was expended in optimizing the shape of the bubble launch tube. The preferred version was a vertical tube with a very short bend and an aperture located as low as feasible. This launcher design was adapted to the Inconel tubes used with molten glass.

It was observed during the studies with oil that the bubble is actually launched a very short period after closure of the downstream valve. The visual observations were compared with the pressures obtained with the transducers so that pressure changes with time could be used for diagnostic purposes during the studies with molten glass.


Figure 9. Relation Between Bubble Diameter and Supply Pressure
Figure 10. Original (a) and Modified (b) Bubble Launchers
For molten glass, the initial bubble launch sequences are similar, but a substantial difference was noticed involving the bubble release time. After a bubble is released in the melt, capillary action tends to draw a small amount of liquid into the tube creating a plug. In subsequent cycles, bubbles are released after the plug is breached (about one second after the downstream valve is opened). This delay appeared to be advantageous because it facilitated monitoring of the pressure within the launch tube and thus precisely determine the time at which the opening bubble is released.

4.2.3 Bubble Characterization

During the development of a system for the generation of helium bubbles, it became necessary to have a technique for assessing the degree to which bubble sizes could be controlled. From the equations which have been found to represent satisfactorily the rise velocity of gas bubbles in liquids, it was evident that if the viscosity and specific gravity of the liquid are kept constant, which would be the case if the temperature is carefully controlled, the rise velocity is proportional to the square of the bubble diameter. Thus, the degree of size uniformity can be determined from the distribution of rise velocities in the same liquid at constant temperature.

Bubble shapes and diameters were determined photographically with a Nikon 8008 camera. Typical bubbles are shown in Figure 11. In general, the bubbles were approximately spherical in shape, with a pear shape being noticed immediately after launch.

All rise velocity measurements were made visually, by timing the interval between the rise of the bubble past two index lines. This method was not very precise, since at lower viscosities the time intervals were on the order of only one
Figure 11. Helium Bubbles in S-2000 Oil
second. With the development of instrumental measurement methods, the precision is expected to improve by eliminating subjective operator errors. Multiple exposure photography, using the precision timer of the Nikon camera, have also been used to estimate rise velocity, as shown in Figure 11.

The volume of the helium metering chamber was determined by launching bubbles into an inverted burette filled with water. The calculations of the chamber volume were made with the ideal gas law equation, taking into account variations in atmospheric pressure and ambient temperature.

It was subsequently found that the helium detector used for determining the arrival time of the bubble at the surface of the liquid was also useful in estimating the mass of helium in the bubble.

4.3 Pressure Measurements

The accurate dispensing of a desired mass of helium contained in a chamber of known volume requires precise measurements of the gas pressure in the chamber. This was accomplished with a high-precision OMEGA manometer. For computer data acquisition, in-line pressures were obtained with transducers. Calibration curves for the transducers are shown in Figure 12. The analog output from the transducers was fed into an OMEGA WB-AAI-B high-resolution data acquisition card. The analog signals were converted to digital outputs with 16 bit resolution at speeds of 200-225 samples per second and fed to an IBM-compatible PC via a serial port. The software provided for the OMEGA card permitted a continuous display of the line pressures as a function
Figure 12. Calibration of Pressure Transducers
of time. Figure 13 shows the pressure in the launch tube after closure of the downstream valve. The delay of about one second caused by the pressure of a molten glass plug is clearly visible. The upper pressure plateau corresponds to the pressure in the line and also in the metering chamber (since the downstream valve is open).

The lower plateau corresponds to the hydrostatic pressure at the point in the liquid where the bubble is launched. The helium mass of the bubble can be calculated from the chamber volume, ambient temperature, gas pressure before the downstream valve is opened, and line pressure after the downstream valve is opened.

The mass of the helium bubble can also be calculated from the two pressure plateaus described above. This pressure difference is, however, considerably smaller and the bubble mass estimate will be less accurate. The ability to continuously monitor and record the line pressures before and after the launch of the helium bubble provided an excellent means to verify the constancy of bubble mass over a period of several days during which hundreds of bubbles were launched.

Monitoring of the pressure profile also provided a useful diagnostic tool. In Figure 14(a), the plateau previously noticed after opening of the downstream valve is absent and the pressure decays to the lower plateau somewhat slower. Examination of the system showed that a small leak had developed. The helium level profile in Figure 14(b) indicates that two, rather than one, bubbles were released.
Figure 13. Profile of Pressure in Launch Tube During Normal Launch
Figure 14. Profile of Pressure in Launch Tube (a) and He Concentration at Detector (b) During Abnormal Launches

29
4.4 Measurement Of Bubble Rise Velocity

During the preliminary studies with transparent liquids at or near room temperature, the rise velocities were determined visually by measuring with a stop watch the time interval between passage of the bubble in front of two reference lines. The major challenge of this study was found to be the development of a remote method for the measurement of bubble rise velocity.

The approach pursued consisted of determining the exact arrival time of a helium bubble at the liquid-gas interface using a helium detector. Several types of gas detectors are suitable for this purpose: the differential thermal conductivity cell and the mass spectrometer type leak detector are two widely used instruments. A Varian 936-40 portable helium leak detector was used in this study. The equipment arrangement is shown schematically in Figure 15. The helium bubble emerging from the liquid is captured in a bonnet which is partially immersed in the liquid. A nitrogen carrier stream is introduced into the bonnet to flush the helium into a transfer tube which is connected to the helium detector located at a distance of several feet. The arrival of the helium spike is signaled by the helium detector as an increase in output voltage which is proportional to the helium concentration in the nitrogen carrier. By carefully controlling the flow rate of the carrier gas, and thus the travel time of the helium spike between bonnet and detector, it is possible to calculate the exact arrival time of the bubble at the liquid-gas interface. The rise velocity measurement is further simplified by employing two launch tubes which introduce helium bubbles of nearly equal mass but at different levels. Since the spacing of the launch positions is known and the difference between the total time elapsed from the introduction to the detection of the two helium bubbles is measured,
Figure 15. Remote Glass Monitoring Concept
the average rise velocity between the two levels can be calculated by dividing the

time difference by the level difference. The bonnet location and carrier gas

introduction were subsequently modified, to prevent depressing the liquid by

excessive pressurization. The bonnet bottom is kept out of the liquid and air

carrier gas is transported to the detector by suction produced with a metering

pump. The modified system is shown in Figure 16.

The calculational procedure with reference to Figure 15 is as follows:

\[ r = \text{average radius of bubbles 1 and 2 (controlled by bubble generators)} \]

\[ t_1 = \text{time for bubble 1 to traverse distance } L_1 \text{ (measured)} \]

\[ t_2 = \text{time for bubble 2 to traverse distance } L_2 \text{ (measured)} \]

\[ d = \text{difference between levels of bubble launch (known)} \]

\[ L_1 = \text{vertical distance of bubble 1 launch point from liquid-gas interface (varies with liquid level)} \]

\[ L_2 = \text{vertical distance of bubble 2 launch point from liquid-gase interface (varies with liquid level)} \]

\[ \Delta P = \text{differential hydrostatic pressure between bubble launch points (measured)} \]

\[ t = \text{time for bubble 2 to traverse distance } d \text{ (calculated)} \]
General Relation (simplified):

\[ \eta = \frac{k \rho g r^2}{u} \]

where

\[ u = \text{average rise velocity of bubbles 1 and 2 (calculated)} \]

\[ \eta = \text{viscosity of liquid (calculated)} \]

\[ \rho = \text{specific gravity of liquid (calculated)} \]

\[ g = \text{gravitational constant (known)} \]

\[ k = 2/9 \text{ in Stoke's formula or } k = 1/3 \text{ in Hadamard-Rybczynski formula.} \]

But

\[ t = t_2 - t_1 \]

\[ v = \frac{d}{t_2 - t_1} \]

\[ \rho = \frac{\Delta P}{gd} \]

so

\[ \eta = \frac{k \left( \frac{\Delta P}{gd} \right) r^2}{\frac{d}{t_2 - t_1}} = \frac{k(\Delta P) r^2 (t_2 - t_1)}{d^2} \]
Figure 16. Carrier Gas Delivery by Suction Method
In the absence of factors which delay the prompt "bursting" of the bubble at the liquid-air interface, the remote bubble detection method based on the helium spike in a carrier gas worked very well. However, when the liquid exhibited high surface tension or when a crust or other solid materials were present on the liquid surface, irregular delays were encountered before the bubbles burst and this introduced appreciable errors. All alternate bubble detection method, based on the change in electrical conductivity or dielectric constant which occurs when a bubble passes between electrodes placed in the liquid, will be used in the future to overcome the bursting delays observed in some experiments.

5. EQUIPMENT

Cannon-Fenske Viscometers

A set of seven viscometers, covering a range of kinematic viscosities from 3 to 100,000 centistokes, was obtained from Cannon Instruments Co., State College, PA. The viscometers were furnished with Certificates of Calibration which gave the viscometer constants at 40 and 100°C. Constants for other temperatures were obtained by interpolation.

An agitated oil bath, provided with electric heaters and cold water coils, allowed temperature control to ± 0.1°C. Excellent agreements were obtained by determining the viscosities of standard oils provided by Cannon Instrument Co.
Helium-Leak Detector

A Varian Leak Detector, Model 936-40 was used. A sampling valve with micrometer adjustment allowed the withdrawal of a small gas sample from the carrier line containing the helium spike. Initially, the sampling valve would be plugged up frequently by volatile glass components. A porous metal filter installed in the carrier line remedied this problem. The leak detector provided visual, audio, and analog electrical signal outputs. The latter was used with the computerized data acquisition system.

High-Temperature Furnace And Controller

A Lindberg Model No. 56622 High Temperature crucible furnace and a Eurotherm 818 communicating temperature controller were used. The furnace was modified as shown in Figures 17 and 18. Control of the temperature of the molten glass, using Inconel-sheathed K-type thermocouples, was possible to ± 1°C. A vertical temperature profile of molten glass contained in a crucible is shown in Figure 19. All internal metal tubes (bubble launchers and thermocouple tubes) were Inconel. Alumina crucible obtained from Coors were used in most experiments. The alumina and Inconel showed little evidence of corrosion by the molten glass, except at the liquid-air interface where severe attack occurred.

Gas Systems

The bubble generating system is shown schematically in Figure 8. The fast-acting solenoid valves were obtained from Angar Scientific. Flexible hoses and Swagelock fittings were used to connect the
Figure 17. High-Temperature Furnace and Test Setup
Clearance for 10-32

- Launch Tubes
- Thermocouple
- Sampling Tube
- Observation Holes

Figure 18. Layout of Penetrations in Cover of High-Temperature Furnace
Figure 19. Temperature Profile Inside Crucible
metering manifold and the launcher tubes. The latter were made of glass, copper, or Inconel. Optimization of the launch tube shape resulted from a systematic investigation of several types. An Emerson air pump was used to provide the necessary suction for the air carrier stream.

**Tempered Water Bath**

To maintain a constant temperature during the BRV measurements with oils and glycerol, a Fisher Model 9000 circulator was used for water contained in a glass tank. Temperature control of ± 1°C was achieved in the temperature range of 5-60°C.

**Camera**

A Nikon 8008 camera was used for characterizing the He bubbles and for the estimation of bubble rise velocities using the precision timer of the camera.

**Gas Flow Controller**

A Vacuum General Model UC2-21 precision gas mass flow controller was used during the metal development studies. While the slow controller operated satisfactorily, the bubble generation method based on timed flow control was abandoned in favor of the metered mass method.

A rotameter was used to control the flow rate of the air carrier stream.
An NCR PC/AT compatible computer was used. It contains a switchable 6/10 MHz 80286 processor, 640 KBytes main memory and 512 KBytes expanded memory. Magnetic storage is provided by a 40 MByte Seagate disc drive partitioned into a 15 MByte disc, exclusively for laboratory software, and a 25 MByte disc for more general utility programs. Two 5-1/4" floppies, one a 1.2 MByte high-density drive, the other a 360 Kbyte drive, are also available. Two serial ports provide access to a campus network and communication with the furnace controller. A parallel port drives an Epson FX-86e dot-matrix printer. Display is provided by a VEGA VGA card and an NEC high-resolution color monitor. These provide moderate speed with excellent display and proved adequate for our experiment.

Signal acquisition and experiment control was provided by an OMEGA WB-AAI-B high-resolution data acquisition card. The card provides 16 analog-to-digital channels and 8 digital input/ouput channels. Analog signals are converted with 16 bit resolution at speeds of up to 225 samples/sec for single channel acquisition (200 samples/sec for multiple channel acquisition).

6. COMPUTER CONTROL, DATA ACQUISITION AND PROCESSING

At the beginning of this study, simple computer programs were used to program the sequence of relay openings and closing. As the study progressed, more sophisticated software was developed which, in addition to controlling the sequence of events, also controlled the acquisition of a large body of data, processed the data to a number of output formats, and also provided a real life graphic display of essential parameters during the course of the experiments.
The latter was extremely useful in providing instantaneous evidence that a He bubble of the proper volume was injected into the molten glass and that a corresponding helium spike was detected in the carrier gas after a reasonable time interval.

6.1 Bubble Generation And Detection

The sequence and approximate time intervals for a complete bubble generation and detection cycle is shown schematically in Figure 20. A bubble generation and detection sequence proceeds as follows:

Immediately after starting the timer, a digital output channel is turned on energizing the SPDT relay which controls the helium flow valves. When the relay is energized, the upstream valve closes and the downstream valve opens, injecting a mass of helium into the launcher tube and connecting tube. After a brief time interval, the digital output channel is turned off, de-energizing the relay which causes the helium flow valve to reset. The Helium bubble is ejected after a very brief time interval.

The arrival of the helium spike in the carrier gas at the helium detector is indicated by an increase in the detector output voltage which is transmitted to the computer. A 100-point running average is calculated to eliminate false signals due to noise spikes. When a rise in helium concentration of 5% above the running average is detected, the timer is read and the total time elapsed between energizing the relay and detection of the helium spike is recorded. The bubble rise time is then calculated by subtracting from the total time the previously determined delay due to bubble breakage and transport of the helium from the bonnet to the detector.
PI - Pressure Indicator
PT - Pressure Transducer
FI - Flow Indicator

<table>
<thead>
<tr>
<th>Time (sec)</th>
<th>Event No.</th>
<th>V-1</th>
<th>V-2</th>
<th>V-3</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>10-2C</td>
<td>switch to Launcher #1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>C</td>
<td>C</td>
<td>10-2C</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>C</td>
<td>C</td>
<td>10-2C</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>4</td>
<td>C</td>
<td>O</td>
<td>10-2C</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>5</td>
<td>C</td>
<td>C</td>
<td>10-2C</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>6</td>
<td>C</td>
<td>C</td>
<td>10-2C</td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>7</td>
<td>C</td>
<td>C</td>
<td>10-2C</td>
<td></td>
</tr>
<tr>
<td>18.5</td>
<td>8</td>
<td>C</td>
<td>C</td>
<td>10-2C</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>9</td>
<td>C</td>
<td>C</td>
<td>1C-20</td>
<td></td>
</tr>
</tbody>
</table>

Figure 20. Bubble Launch Cycle
The frequency at which bubbles can be launched into the liquid is determined by the total time elapsed between energizing the relay and detection of the helium. With the system used in this phase of the study, one bubble can be launched about every thirty seconds.

6.2 Graphic Performance Display

During an experiment, simultaneously with the continuous acquisition and storage of data, a graphic display is provided which provides continuous evidence of proper sequencing and performance of the helium bubble generation and detection. A photograph of such a display is shown in Figure 21. A computer reconstructed display is shown in Figure 22. The length of the upper plateau is an indication of the proper release of the bubble. The difference in the pressures of the upper and lower plateaus is a measure of the bubble size. The deviation of the time interval between the beginning of the lower plateau (i.e. instance of bubble ejection) and beginning of the increase of helium concentration at the leak detector is an indication that the transport of the helium from the glass surface to the detector proceeded normally. The shape of the helium leak detector output is an indication that only a single bubble was ejected. Finally, integration of the detector output signal provides an independent measure of the size of the bubble.

6.3 Computer Software

6.3.1 General

The software for experiment control and data acquisition was written in Microsoft QuickBasic version 3.0. The language is fully structured and allows sophisticated control and display programs to be developed quickly. A
Figure 21. Photographs of Computer Monitor
Figure 22. Graphic Performance Display
weakness in QuickBasic 3.0 is its primitive library support with awkward rules of access to global constants and variables. The access problems were surmountable, however, and the limited library support was used to modularize the software.

The development of the software proceeded in a bottom-up fashion. Individual control and acquisition elements were written and tested as the apparatus to be controlled and monitored was built. Low-level routines, each performing a single basic task, are called by higher-level routines which perform multiple complex tasks. These high-level modules are called by the main executable program to perform all the control and acquisition tasks required by the experiment. The following example, describing the monitoring of the pressure transducer, will illustrate the hierarchy of the software environment: reading a pressure transducer involves issuing a sequence of commands to the Omega data acquisition card to first select the channel which is dedicated to the transducer, then to retrieve an analog signal from that channel. The software commands to perform this sequence were collected into a subroutine named `read.pressure'. Another routine, on a higher level of the code structure, uses this low-level routine to determine the precise time at which a bubble is launched from the launch tube. This routine, named 'find.launch' integrates several low-level routines to perform a single-high-level operation. At each level of the command structure, the routines are modular, that is independent of other routines, to perform a single high-level operation. Thus the high-level routines, 'find.launch' and 'detect.bubble', are independent of one another. This modularity allows several different main programs to be written with different combinations of high-level routines without having to include all the other high-level routines as
Finally, the main experiment program incorporates several of the high-level modules to perform the various individual high-level tasks which constitute a run.

The nature of the molten glass experiment makes direct visual monitoring of the experiment difficult; everything in the furnace glows at virtually the same temperature, offering very little visual contrast. To adequately monitor the experiment, the pressure transducers and a helium detector are used as diagnostic instruments. Graphically displaying the signals from these instruments greatly enhances their diagnostic value. Therefore, software was developed to take advantage of the graphics capabilities of the software and the high-resolution color monitor to allow both, a graphical representation of selected data as well as a standard numerical display of the variables of the experiment. Program listings are given in Appendix B.

6.3.2 A/D Card Driver

The Omega analog-to-digital converter comes with a set of software primitives that issue the actual machine-level instructions to the card. These routines, collected into the object file `bcall.obj,' perform card setup, analog signal acquisition and digital input/output control. The object file has been installed as a library in the file `userlib.exe' and is loaded when QuickBasic is initiated. One weakness of these primitives is that values returned from an analog conversion are not returned in QuickBasic format. Numbers must be converted explicitly to be meaningful. This slows down the execution speed of the program.
6.3.3 **Low-Level Routines**

**Init.cards**

Before analog voltages can be converted and digital output can be performed, the Omega card must be properly initialized. The routine `init.card` issues the commands to set the number of analog channels available and their ranges. Analog input channels 0 through 3 are set to a range of -5 to +5 volts. Analog input channel 4 is calibrated for a K-type thermocouple. Also set is the delay between readings of consecutive channels during a multiple channel read. A setting of 300 corresponds to a delay of 9 msec. The card contains its own precision voltage reference against which the analog input channels are calibrated. The calibration occurs after all other parameters for the analog channels have been set. The digital channels 0 through 2, which control the launch valves, are configured to be output channels. Their status is set to high which corresponds to the valves being closed. There are no parameters sent to or returned by the routine.

**Make.bubble**

The launching of a bubble is initiated by the opening and closing of a pair of relay-activated valves each of which is controlled by a single digital I/O channel. The program statements which control this sequence are collected into a single low-level subroutine called `launch-bubble`.

The parameter sent to the routine is ‘bubbler’ which selects the launch tube into which a gas flow will be directed. ‘Start.time,’ returned by ‘make.bubble’ is the time at which the downstream valve is opened. This time is considered to be the time at which the bubble emerges if no plug exists to delay bubble launch.
Read.helium

The concentration of helium in the sampling stream is represented by the helium detector as a proportional analog voltage. This voltage ranges from zero to -6 volts, zero representing the minimum helium concentration. This voltage is read by the routine 'read.helium', which makes a single conversion of the channel into which the helium detector output is directed. 'Convert' is called to convert the voltage to IEEE format and the number is then sign reversed to yield a positive value.

The parameters returned by the routine are 'level', the measured helium concentration, and 'time' the time at which the conversion took place.

Read.pressure

The pressure within each launch tube is determined by an Omega PX94 005G5V pressure transducer. Output signals from 1 to 6 volts correspond to pressures from 0 to 5 psi, respectively. The routine 'read.pressure' initiates a single conversion of the proper channel. While the conversion to true voltage units is made automatically by the card driver, the conversion to pressure units is made explicitly in the routine. The pressure transducers have been calibrated with a precision analog gauge to arrive at a linear fit of the transducer readings to the gauge readings. The constants which describe this linear fit are incorporated into the routine and are used to convert the voltage output from the transducers into real pressure values.
The parameter sent to 'read.pressure' is 'bubbler' which selects in which launch tube the pressure is to be read. Parameters returned from the routine are the measured pressure, 'pressure.value', and the time at which the measurement was made, 'time'.

6.3.4 High-Level Modules

Find.launch

Precise determination of the time at which a bubble is released from a launch tube is essential for accurate BRV measurements. The routine 'find.launch' is responsible for making this determination by monitoring the pressure in the launch tube. After the downstream valve is opened, a small amount of time is required to push the small glass plug aside to allow the release of a bubble.

Detect.bubble

This routine is used to signal the arrival of the helium spike at the detector; the time is registered and a new cycle is initiated.

6.3.5 Graphics Package

Graphical display of the data is essential to properly monitor the performance of the bubble launch and detection system. While QuickBasic 3.00 provided the extensive tools for the display of graphical information, a graphics library was adapted, using these tools, to more easily implement the graphical display. The variables which define the screen limits and coordinate limits, as well as the lines on which text is printed, are defined in a common block to allow access to them.
throughout the program environment. The routines `set.physical.coords,' 'set.text,' and `set.usr.coords' establish the values of this common block.

6.4 Estimation Of Glass Composition

The program TERNARY written in FORTRAN is used to estimate the composition of a ternary system from two physical properties. Experimentally determined property values at a given temperature are plotted on a ternary diagram (or on a simplified two-dimensional plot) and lines of constant property values are drawn on the diagram by interpolation between the experimental points. Polynomial equations are fitted to the lines by the least-square method. Specific compositions which correspond to the roots of the equations representing the pairs of property values are then obtained, using the Newton-Raphson Method. A listing of the program is provided in Appendix A.

The program was tested with data obtained from the literature for sodium borosilicate glasses. Shown in Table 3 are the compositions for which the viscosity and density data were obtained and the corresponding compositions estimated by the computer program

<table>
<thead>
<tr>
<th>COMPOSITION TAKEN</th>
<th>COMPOSITION CALCULATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>mol%</td>
<td>mol%</td>
</tr>
<tr>
<td>B₂O₃   SiO₂</td>
<td>B₂O₃   SiO₂</td>
</tr>
<tr>
<td>20.0   61.9</td>
<td>19.99   62.01</td>
</tr>
<tr>
<td>25.0   61.0</td>
<td>24.96   58.95</td>
</tr>
<tr>
<td>6.0    76.3</td>
<td>5.29    76.42</td>
</tr>
<tr>
<td>2.0    80.7</td>
<td>1.85    80.69</td>
</tr>
<tr>
<td>21.7   69.4</td>
<td>22.21   68.68</td>
</tr>
<tr>
<td>11.3   81.0</td>
<td>11.36   81.02</td>
</tr>
</tbody>
</table>

Table 3. Test of Computer Program TERNARY
7. RESULTS

7.1 Cannon-Fenske Viscosity Measurements

The results obtained by the Cannon-Fenske method for glycerol and S-2000 oil served as the reference to validate the determinations by the BRV method, an essential step in the development of remote monitoring concept. The results are shown in Figures 4 and 5.

7.2 Bubble Size Determinations

7.2.1 Photographic Method

Brief exposure time photographs of bubbles rising in oil were taken with the Nikon 8008 camera. A typical photograph is shown in Figure 11. By comparison with a scale placed in the vicinity of the bubble trajectory, the diameter of the bubble was obtained by direct linear measurement. This method was time-consuming and was used during the initial steps to establish an approximate relation between supply gas pressure and bubble size.

7.2.2 Pycnometric Method

Accurate measurement of bubble size were obtained by a pycnometric, water displacement method.

Bubbles generated by the system shown in Figure 8 were introduced into an inverted burette filled with water and with the open end placed in a beaker also filled with water. The bubble volumes at standard temperature and pressure were obtained from the burette readings, corrected for temperature, barometric pressure, and water head.
Results of a calibration run are shown in Figure 23. These results were deemed to be adequate for basing all subsequent bubble volume (or diameter) estimates on the supply gas and hydrostatic pressures.

The data obtained in these calibrations were also used to calculate the volume of the helium metering chamber, the section of pipe between the two solenoid valves.

7.2.3 Pressure Measurement Methods

The adequacy of the method based on the determination of the supply and hydrostatic pressures was verified as described above. It developed in the course of the study that the pressure in the launch tubes could be conveniently determined with the transducers and recorded by the computer. It remained to be shown that a reproducible relation existed between the differences in launch tube pressures just before and after ejection of the bubble and the helium supply pressure. The results shown in Figures 24, 25, and 26 show that the launch tube pressure difference (LTPD) is indeed a useful parameter for estimating bubble size. It is also noteworthy that, as shown in Table 4, the LTPD values were not affected by changes in glass temperature.

7.2.4 Integration Of Helium Concentration Method

As described in section 6.2, it developed that the bubble size could also be estimated by integrating the leak detector output signal triggered by the arrival of the helium spike in the carrier gas.
Figure 23. Determination of Bubble Volume (a) and Diameter (b) by Pycnometry
**Figure 24.** Launch Tube Pressure Differences as a Function of Supply Pressure for SRL Waste Glass
Temperature = 25°C
Viscosity (from supplier) = 55.21 poise

Figure 25. Launch Tube Pressure Differences as a Function of Supply Pressure for S-2000 Oil
Temperature = 25°C
Viscosity (from supplier) = 238.2 poise

Figure 26. Launch Tube Pressure Differences as a Function of Supply Pressure for S-8000 Oil
Figure 27 gives the results of an experiment in which the helium supply pressure was varied. The corresponding helium detector output curves were integrated by the computer. This method showed promise as a backup method for monitoring bubble size. It is quite essential that the flow rate of the air carrier stream be carefully regulated and that other variables affecting the operation of the leak detector be also held constant.
Figure 27. Determination of Bubble Size by He Signal Integration
<table>
<thead>
<tr>
<th>Supply Pressure psig</th>
<th>Temperature °C</th>
<th>(LTPD) upper psi</th>
<th>(LTPD) lower psi</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.5</td>
<td>1024</td>
<td>0.16 (14)</td>
<td>0.17 (13)</td>
</tr>
<tr>
<td>8.0</td>
<td>1040</td>
<td>0.18 (32)</td>
<td>0.16 (39)</td>
</tr>
<tr>
<td>8.0</td>
<td>1050</td>
<td>0.18 (34)</td>
<td>0.16 (35)</td>
</tr>
<tr>
<td>7.9</td>
<td>1062</td>
<td>0.18 (35)</td>
<td>0.16 (34)</td>
</tr>
<tr>
<td>8.0</td>
<td>1092</td>
<td>0.18 (26)</td>
<td>0.16 (27)</td>
</tr>
</tbody>
</table>

Numbers in parentheses indicate the number of individual data points from which the average LTPD values were obtained.

Table 4. Effect of Temperature on LTPD Values
7.3 Bubble Rise Velocity

Under laboratory conditions were the level of glass in a crucible can be determined accurately, the BRV could be determined with a single bubble launcher. In an actual melter where the glass level varies, the BRV is best determined by using two bubble launch tubes located at different elevations.

The results in Figure 28 show that at a constant supply pressure the gas traveling times decrease with rising temperature, in accordance with the expected decrease in the viscosity of the liquid.

The results in Figure 29 indicate that the gas traveling time at a constant temperature decrease with a rise in supply pressure, i.e., with increasing bubble size, as would be expected from Stokes' equation.

The experimental data for the BRVs for SRL waste glass as a function of temperature and supply pressure are shown in Figures 30 and 31. This type of data can be used for the development of empirical equations for the conversion of BRV data to dynamic viscosities.

In Figure 32, experimental BRVs obtained for S-2000 oil at several temperatures are plotted along with BRVs calculated using the Hadamard-Rybczynski (H-R) formula. There appears to be a systematic bias between the two sets, suggesting that a modification of the H-R formula would be necessary.

To test the overall utility of the BRV method used in an entirely remotely operated mode, a reduced properties plot was prepared as described for oil in section 3.2.2. As seen in Figure 33, there is excellent agreement between the BRVs and the viscosities reported for SRL waste glass.
Figure 28. Gas Traveling Times as a Function of Temperature
Temperature = 1050°C

- upper bubbler
- lower bubbler

Figure 29. Gas Traveling Times as a Function of Supply Pressure
Figure 30. Bubble Rise Velocity as a Function of Temperature for SRL Glass (Samples 3T-1, 3B, 4T-1/5-28-875/5)
Temperature = 1050°C

Figure 31. Bubble Rise Velocities as a Function of Supply Pressure for SRL Glass (Samples 3T-1, 3B, 4T-1/5-28-875/5)
Figure 32. Experimental and Calculated (H-R) BRVs for S-2000 Oil as a Function of Temperature

Supply Pressure = 10 psig
Figure 33. Comparison of Reduced Reciprocal BRVs and Reduced Viscosities for SRL Waste Glass
To overcome the occasional problems encountered with delays in the "bursting" of the bubble due to large surface tension of the liquid, an alternate bubble detection method was tested. This method is based on the change in electrical conductivity which occurs when a bubble passes between two electrodes placed in the liquid. In this test helium bubbles were generated in a glycerol solution containing lithium chloride. An AC potential of 3-5 volts was applied across two aluminum electrodes which formed one branch of a Wheatstone bridge. Passage of the bubble produced a sharp transient in the bridge current. The test setup is shown in Figure 34.

8. FUTURE WORK

The "proof-of-principle" phase having been successfully completed, the following tasks are recommended for future work:

- Development of a gas bubble detection method based on the generation of an electrical signal.

- Correlation of BRV data and viscosities obtained by conventional methods to produce an empirical equation for the conversion of BRV data to dynamic viscosities.

- **Design** of an integrated probe compatible with the SRL melter. (One concept is shown in Figure 35.)

- Testing of an optimized probe, first with a high-viscosity liquid at moderate temperatures and then with SR waste glass.

- Compilation of a database containing the compositions, viscosities, and densities of both acceptable and non-acceptable SR waste glasses.
Figure 34. Bubble Detection by Electrical Signal
Figure 35. Concept for an Integrated BRV Probe
- Upgrading of the computer programs which would provide glass composition estimates (in terms of the three major components) from the monitored parameters (temperature, density, and viscosity).

- Applicability of the BRV method to the remote characterization of other materials used in the DWPF (e.g., slurries).

9. CONCLUSIONS

This project has successfully demonstrated a novel concept for the near-continuous monitoring of molten glass. The following elements basic to the concept have been verified experimentally or were derived from information available in the literature:

Existence of a direct correlation between the rise velocity of gas bubbles in a liquid and the viscosity of the liquid.

Availability of a method for the remote detection of a gas bubble emerging from a liquid surface.

Availability of a method for the remote determination of the density of a liquid.

Availability of computer hardware and software to control the sequence of events in the measurements of bubble rise velocity, temperature, and density; acquisition and processing of the data; and final qualification of the glass by reference to an empirical data bank.
Suitability of components to be placed inside the melter to withstand elevated temperatures, high radiation fields, and corrosive attack by the liquids and gases present.

Availability of physical properties and calculational method for estimating the empirical composition of a glass by determining the temperature, the density, and the viscosity of the molten glass.

All the objectives listed in the Statement of Work for this project have been met.
APPENDIX A

Computer Program TERNARY (FORTRAN V) calculates the composition of borosilicate glass from density and viscosity at 1100°C.
Flow Diagram of TERNARY

a. Main Program

START

READ
density & viscosity

CALL COMP
calculate the composition
of Borosilicate Glass

WRITE
density, viscosity
& composition

END
b. Subroutine COMP( DEN, VIS, XI, YI )

1. READ composition data sets for density & viscosity

2. SUBROUTINE CURVE2
   CALL CURVE2 calculate the coefficients of the polynomial from the curve which has the given value of the density or viscosity

3. WRITE the coefficients

4. FIND the compositions of the point on which one curve intersects the other

   IF the point is within the data range
     yes
     PRINT the compositions of Borosilicate Glass
     Return
   else
     No
     PRINT "out of range"

5. [Return]
c. Subroutine CURVE2(PTY, PR, L, M, N, X, Y, J, BR)

- Locate some required value, PTY within the range between two values, PR's of the property

- Interpolate the coefficients of the polynomial equation from two sets of coefficients on the curve within the range

RETURN

d. Subroutine LEAST( N, M, X, Y, B)

- Generate elements of the matrix from Least-square Method

- Invert matrix A to B

- Calculate the coefficients of the polynomial equation for the curve

RETURN
Program Listing of TERNARY

NOFLOATCALLS
STORAGE:2
DEBUG

C
C PROGRAM TERNARY
C
C CALCULATE POSSIBLE COMPOSITIONS OF THE TERNARY SYSTEM
C WITH ITS SELECTIVE PHYSICAL PROPERTIES
C ( DENSITY & VISCOSITY )
C
C PROGRAM TERNARY
open(unit=5,file='curin')
open(unit=6,file='curout')
WRITE(*,1205)
WRITE(6,1205)
50 READ(5,1200) DEN,VIS
IF(DEN.EQ.0.0) STOP
C ** DEFINE THE RANGE OF INPUT VALUES
C IF((DEN.LT.2.15).OR.(DEN.GT.2.50)) GO TO 600
IF((VIS.LT.2.0).OR.(VIS.GT.4.0)) GO TO 700
WRITE(6,1210) DEN,VIS
C CALL COMP(DEN,VIS,SOD,SI)
C GO TO 50
WRITE(6,1250)
600 WRITE(*,1250)
GO TO 50
WRITE(6,1260)
700 WRITE(*,1260)
1250 FORMAT(/1H1,5(/),20X,' *** DENSITY IS OUT OF THE RANGE ***')
1260 FORMAT(/1H1,5(/),20X,' *** VISCOSITY IS OUT OF THE RANGE ***')
800 GO TO 50
C ** INPUT OR OUTPUT FORMAT
C
1200 FORMAT(2(F10.2))
1205 FORMAT(/15X, '****** SEARCH # FROM MAXIMUM B2O3 CONC. = 1'//
* 15X,'****** SEARCH # FROM MINIMUM B2O3 CONC. = 2'//
*/5X,'DENSITY',3X,'LOG-VISCOSITY',7X,'SEARCH '
* 4X,'B2O3',*5X,'SIO2'/5X,'(G/CC)',6X,'(POISE)',23X,
* '(MOL%)',3X,'(MOL%)'//)
1210 FORMAT(/6X,F6.2,6X,F6.2)
1300 FORMAT(20X,'B2O3 MOL% = ',F6.2//20X,'SIO2 MOL% = ',F6.2/)
** CALCULATE THE CHEMICAL COMPOSITION OF THE POINT WHERE THEIR CURVES CROSS FROM THE VALUES OF VISCOSITY AND DENSITY

** SUBROUTINE COMP(DEN,VIS,XC,YC)

** 5 VISCOSITY CURVES (LV)
** 8 DENSITY CURVES (LD)
** EACH VISCOSITY CURVE FITS '5-TH' DEGREE POLYNOMIALS (MV)
** EACH DENSITY CURVE FITS '3-RD' DEGREE POLYNOMIALS (MD)
** ( THE DEGREES DETERMINED FROM INDEPENDENT PROGRAM IN APPENDIX B. )
** 6 DATA POINTS CHOSEN FROM EACH VISCOSITY CURVE (NV)
** 6 DATA POINTS CHOSEN FROM EACH DENSITY CURVE (ND)

DATA LV/5/,LD/8/,MV/3/,MD/5/,NV/6/,ND/6/

** VISCOSITY VALUES

DATA (VS(J),J=1,5)/2.0,2.5,3.0,3.5,4.0/

** PERCENT 'B2O3' IN THE GLASS WHICH VISCOSITY VALUE IS SHOWN ABOVE

DATA ((XV(I,J),I=1,6),J=1,5)
* /23.6,20.0,17.3,14.4,12.0,10.4
*,/24.0,21.2,15.4,9.7,5.8,2.4
*,/24.3,19.4,10.5,6.0,3.5,0.0
*,/25.0,21.5,18.0,12.8,5.0,0.0
*,/18.0,15.3,12.0,7.7,4.8,0.8/

** PERCENT 'SiO2' IN THE GLASS WHICH VISCOSITY VALUE IS SHOWN ABOVE

DATA ((YV(I,J),I=1,6),J=1,5)
* /60.0,62.0,63.0,63.8,64.5,65.0
*,/62.6,64.8,68.0,70.3,71.3,69.9
*,/67.5,70.6,75.0,76.5,76.5,74.5
*,/71.4,73.2,75.0,77.2,80.0,81.0
*,/78.2,79.4,80.8,82.3,83.2,84.2/

** DENSITY VALUES
DATA
(DN(J),J=1,8)/2.15,2.20,2.25,2.30,2.35,2.40,2.45,2.50/

** PERCENT 'B2O3' IN THE GLASS WHICH DENSITY VALUE IS SHOWN ABOVE

DATA ((XD(I,J),I=1,6),J=1,8)
*/40.5,36.0,34.4,30.4,28.0,25.0
*,38.1,32.4,28.5,24.0,17.2
*,36.5,31.1,22.6,18.0,15.0,11.6
*,35.1,29.5,25.5,21.0,12.2,5.7
*,33.7,23.4,15.4,8.4,5.6,1.2
*,32.3,17.4,10.2,7.6,4.8,1.0
*,30.7,24.2,15.4,9.4,5.4,0.4
*,26.9,21.6,17.4,11.1,3.1,0.0/
*,14.7,14.3,11.6,7.0,4.0,2.1/

** PERCENT 'SiO2' IN THE GLASS WHICH DENSITY VALUE IS SHOWN ABOVE

DATA ((YD(I,J),I=1,6),J=1,8)
*/51.9,58.0,60.0,65.0,67.5,71.6
*,51.9,60.0,65.0,70.0,75.0,79.0
*,51.9,60.0,70.0,74.5,78.0,83.8
*,51.9,60.0,65.0,70.0,80.0,87.5
*,51.9,65.0,74.6,82.8,84.4,83.4
*,51.9,70.0,78.0,80.0,81.0,79.0
*,51.9,60.0,70.0,74.6,76.4,74.0
*,51.9,60.0,65.0,69.5,66.1,65.0/
*,50.9,51.9,54.0,53.0,52.6,52.5/

CALL COEFF(DEN,DN,LD,MD,ND,XD,YD,BD)

CALL COEFF(VIS,VS,LV,MV,NV,XV,YV,BV)

WRITE(6,1001) (BV(I,1),I=1,4)

WRITE(6,1002) (BD(I,1),I=1,6)

1001 FORMAT('//2X,'VISCOUSITY COEFF. : ',/5X,4(E12.5))

1002 FORMAT('//2X,'DENSITY COEFF. : ',/5X,6(E12.5))

** INTERPOLATE each COEFFICIENT OF POLYNOMIAL EQN
** BY THE CORRESPONDING COEFF. PAIR OF TWO NEAREST CURVES

DO 10 I =1,6
10 A(I)=BD(I,1)-BV(I,1)

ITER=1
ICOUNT=1
XST=20.

B=A(6)

CALL COEFF(DEN,DN,LD,MD,ND,XD,YD,BD)

CALL COEFF(VIS,VS,LV,MV,NV,XV,YV,BV)

WRITE(6,1001) (BV(I,1),I=1,4)

WRITE(6,1002) (BD(I,1),I=1,6)

1001 FORMAT('//2X,'VISCOUSITY COEFF. : ',/5X,4(E12.5))

1002 FORMAT('//2X,'DENSITY COEFF. : ',/5X,6(E12.5))

** INTERPOLATE each COEFFICIENT OF POLYNOMIAL EQN
** BY THE CORRESPONDING COEFF. PAIR OF TWO NEAREST CURVES

DO 10 I =1,6
10 A(I)=BD(I,1)-BV(I,1)

ITER=1
ICOUNT=1
XST=20.

B=A(6)

CALL COEFF(DEN,DN,LD,MD,ND,XD,YD,BD)

CALL COEFF(VIS,VS,LV,MV,NV,XV,YV,BV)

WRITE(6,1001) (BV(I,1),I=1,4)

WRITE(6,1002) (BD(I,1),I=1,6)

1001 FORMAT('//2X,'VISCOUSITY COEFF. : ',/5X,4(E12.5))

1002 FORMAT('//2X,'DENSITY COEFF. : ',/5X,6(E12.5))

** INTERPOLATE each COEFFICIENT OF POLYNOMIAL EQN
** BY THE CORRESPONDING COEFF. PAIR OF TWO NEAREST CURVES

DO 10 I =1,6
10 A(I)=BD(I,1)-BV(I,1)

ITER=1
ICOUNT=1
XST=20.

B=A(6)

CALL COEFF(DEN,DN,LD,MD,ND,XD,YD,BD)

CALL COEFF(VIS,VS,LV,MV,NV,XV,YV,BV)

WRITE(6,1001) (BV(I,1),I=1,4)

WRITE(6,1002) (BD(I,1),I=1,6)

1001 FORMAT('//2X,'VISCOUSITY COEFF. : ',/5X,4(E12.5))

1002 FORMAT('//2X,'DENSITY COEFF. : ',/5X,6(E12.5))

** INTERPOLATE each COEFFICIENT OF POLYNOMIAL EQN
** BY THE CORRESPONDING COEFF. PAIR OF TWO NEAREST CURVES

DO 10 I =1,6
10 A(I)=BD(I,1)-BV(I,1)

ITER=1
ICOUNT=1
XST=20.
30 CONTINUE
   B=A(2)+XST*B
   R1=A(1)+XST*B
   R2=B+XST*C
   RR=R1/R2
606 FORMAT(//20X,'ITER =',I3,5X,'RATIO =',E12.5)
   IF(ITER.GT.100) GO TO 35
   ITER=ITER+1
   XST=XST-RR
   IF(ABS(RR).LT.2.E-01) GO TO 40
   GO TO 15
35 PRINT 400
40 XC=XST
   IF((XC.GT.40.0).OR.(XC.LT.0.0)) THEN
      WRITE(6,5000)ICOUNT
      GO TO 95
   END IF
   YC=BD(6,1)
   DO 90 I=1,5
      K=6-I
      YC=BD(K,1)+XC*YC
   WRITE(6,5010)ICOUNT,XC,YC
95 IF(ICOUNT.GE.2) GO TO 100
   ICOUNT=ICOUNT+1
   XST=0.0
   GO TO 15
5000 FORMAT(35X,I5,7X,'NO ROOT FOUND ')
5010 FORMAT(35X,I5,5X,F6.2,4X,F6.2)
400 FORMAT(19H FAILED TO CONVERGE)
100 RETURN
END

C ** CALCULATE THE COEFFICIENTS OF POLYNOMIAL EQUATION OF
C ** THE CURVE
C ** FROM THE NUMERICAL VALUE OF PHYSICAL PROPERTY

SUBROUTINE COEFF(PTY,PR,L,M,N,X,Y,BR)
DIMENSION B(30,30),BR(30,30),BL(30,30),X(30,30)
   ,Y(30,30),XP(30),YP(30,30),PR(10)
DATA LJ/1/
   MP1=M+1
   PL=0.0
   DO 20 J=1,L
      PH=PR(J)
      KK=J
      IF(PH-PTY) 15,25,30
   15 PL=PH
   20 CONTINUE
   DO 28 I=1,N
      XP(I)=X(I,KK)
   28 YP(I,1)=Y(I,KK)
C ** FIND THE COEFFICIENTS OF NEW CURVE FROM INTERPOLATED
C DATA POINTS
C
CALL LEAST(N,M,LJ,XP,YP,BR)
RETURN

30 DB=PH-PL
DS=PTY-PL
RT=DS/DB
DO 55 K=1,2
DO 40 I=1,N
IJ=KK-K+1
XP(I)=X(I,IJ)
40 YP(I,1)=Y(I,IJ)
   CALL LEAST(N,M,LJ,XP,YP,B)
   IF(K.EQ.2) GO TO 50
   DO 45 J=1,LJ
   DO 45 I=1,MP1
45 BL(I,J)=B(I,J)
55 CONTINUE
50 DO 57 J=1,LJ
   DO 57 I=1,MP1
57 BR(I,J)=(BL(I,J)-B(I,J))*RT+B(I,J)
100 FORMAT(//10X,'THE PROPERTY DATA IS ON ONE OF THE
   *CURVES',//)
RETURN
END

C ** CALCULATE THE COEFFICIENTS OF POLYNOMIAL EQUATION OF
C THE CURVES
C **
C      GIVEN AS INPUT DATA
C
C SUBROUTINE LEAST(N,M,L,X,Y,B)
C
DIMENSION A(30,30),B(30,30),C(30,30),X(30)
*,Y(30,30),W(30)
DATA (W(K),K=1,9) /.1.,1.,1.,1.,1.,1.,1.,1./
DO 30 I=1,N
C(I,1)=1.0
30 CONTINUE
MP1=M+1
DO 35 J=2,MP1
DO 35 I=1,N
35 C(I,J)=C(I,J-1)*X(I)
   DO 40 I=1,MP1
   DO 40 J=1,MP1
   A(I,J)=0.0
   DO 40 K=1,N
40 A(I,J)=A(I,J)+C(K,I)*C(K,J)*W(K)
   DO 45 J=1,L
   DO 45 I=1,MP1
   B(I,J)=0.0
   DO 45 K=1,N
45 B(I,J)=B(I,J)+C(K,I)*Y(K,J)*W(K)
   CALL MATINV(A,MP1,B,L,DET)
**CALCULATE THE INVERSE MATRIX TO SUPPORT THE LEAST SQUARE METHOD**

```fortran
SUBROUTINE MATINV(A,N,B,M,DET)
  DIMENSION A(30,30),B(30,30),IPVOT(30)
  *,INDEX(30,2),PIVOT(30)
  COMMON IPVOT,INDEX,PIVOT
  EQUIVALENCE (IROW,JROW),(ICOL,JCOL)
  DET=1.
  DO 17 J=1,N
  IPVOT(J)=0
  DO 135 I=1,N
    T=0.
    DO 9 J=1,N
      IF(IPV0T(J)-1)13,9,13
      DO 23 K=1,N
        IF(IPVOT(K)-1)43,23,81
        IF(ABS(T)-ABS(A(J,K))) 83,23,23
        IROW=J
        ICOL=K
        T=A(J,K)
      23 CONTINUE
    9 CONTINUE
    IPV0T(ICOL)=IPV0T(ICOL)+1
    IF(IROW-ICOL) 73,109,73
    DET=-DET
    DO 12 L=1,N
      T=A(IROW,L)
      A(IROW,L)=A(ICOL,L)
    12 A(ICOL,L)=T
    IF(M) 109,109,33
    INDEX(I,1)=IROW
    INDEX(I,2)=ICOL
    A(IROW,I)=A(ICOL,I)
    12 A(ICOL,I)=T
    IF(M) 109,109,33
    DET=DET*PIVOT(I)
    A(ICOL,ICOL)=1.
    DO 205 L=1,N
      A(ICOL,L)=A(ICOL,L)/PIVOT(I)
    IF(M) 347,347,66
    DO 52 L=1,M
      B(ICOL,L)=B(ICOL,L)/PIVOT(I)
    DO 135 LI=1,N
      IF(LI-ICOL) 21,135,21
      T=A(LI,ICOL)
      A(LI,ICOL)=0.
    DO 89 L=1,N
  END
```
89 A(LI,L) = A(LI,L) - A(ICOL,L) * T
   IF(M) 135, 135, 18
18 DO 68 L = 1, M
68 B(LI,L) = B(LI,L) - B(ICOL,L) * T
135 CONTINUE
222 DO 3 I = 1, N
      L = N - I + 1
      IF(INDEX(L,1) - INDEX(L,2)) 19, 3, 19
19 JROW = INDEX(L,1)
      JCOL = INDEX(L,2)
      DO 549 K = 1, N
      T = A(K, JROW)
      A(K, JROW) = A(K, JCOL)
      A(K, JCOL) = T
549 CONTINUE
3 CONTINUE
81 RETURN
END
APPENDIX B

Programs for experimental control, data display, data analysis, and statistical evaluation of data. All programs in QUICK-BASIC.

NBUBBLES.BAS
CONTRLIB.BASS
INTEGRAT.BAS
MANYBUBB.BAS
CALIB_PT.BAS
FIT_LINE.BAS
PLOTIT.BAS
PLOTPACK.BAS
FURNACE.BAS
FURNLIB.BAS
STATSLIB.BAS
SELTMP.BAS
BAROMETR.BAS
DENSSTAT.BAS
DELASTAT.BAS
KEEPTEMP.BAS
' Visc.Bas
' 10/4/89
' Program controls the generation and detection of helium bubbles in
' a viscous liquid. Assuming the proper interpretation of the various
' components of the time delay between generation and detection, the
' program can calculate the viscosity of the liquid using previously
' determined calibration constants.
' Helium concentration in sampling stream is read from an analog channel,
' and a bubble is considered to have been detected when a rise of greater
' than a specified threshold is seen in the concentration.
' A pressure transducer for each bubbler will be monitored in the future as
' analog input channels #1 and #2 to determine the hydrostatic pressure and
' hence the density of the liquid. The pressure profile of each launcher
' also determines when a bubble has been launched: a rapid pressure drop
' from a high pressure plateau.
' Furnace temperature will be monitored and
' controlled via the digital interface with the furnace controller. The
' integrated Platinel thermocouple will also be monitored to insure the inner
' furnace temperature does not significantly exceed the 1200 degree C maximum
' temperature rating for the furnace.
' Either bubble rise velocity and density statistics can be displayed or
' graphs of launcher pressure and helium level. Command keys allow toggling
' between the different views.

CONST DEPTH.DIFF = 4.29  ' Difference in depth of launchers (cm)
CONST PURGETIME = 10  ' Time after bubble is detected to purge He detector.
CONST MAXPRINTLINE = 54  ' Maximum number of lines per printed page
CONST DATAFILENUM = 1  ' File number for data
CONST VERSION = "ver. NOV. 13, 1989"  ' Identifies last program revision

REM *INCLUDE: 'D:ROBRSOURCE\BUBLIB.BAS'

printline = 60  ' Keeps track lines printed to log file.
reset.flag = FALSE  ' Flag will cause reset of running averages if TRUE
graph = TRUE

' Open data file
INPUT "Enter data file name: ", datafile$  
CHDIR "D:ROBRDATA"
OPEN datafile$ FOR OUTPUT AS #DATAFILENUM LEN = 1

' Initialize the data acquisition and output and digital i/o cards
CALL init.cards

' Initialize furnace communications.
CALL monitor.furnace(temperature)

CALL set.screen(graph)

' This is the main control and acquisition loop
DO
   delta.pressure = 0
   FOR bubbler = 1 TO 2
CALL command.line(graph,bubbler)

launchtime = -1
DO WHILE launchtime = -1
    CALL make.bubble(bubbler,启动时间)
    CALL find.launch(launchtime, pressure, bubbler, graph)

    ' Get any command flags that have been issued.
    comkey$ = INKEY$

    IF comkey$ = "g" OR comkey$ = "G" THEN graph = TRUE: CALL set.screen(graph) ' Turn on graphing
    IF comkey$ = "i" OR comkey$ = "I" THEN graph = FALSE: CALL set.screen(graph) ' Turn off graphing
    IF comkey$ = "r" OR comkey$ = "R" THEN reset.flag = TRUE ' Reset running averages
    IF comkey$ = "q" OR comkey$ = "Q" THEN
        IF launchtime = -1 THEN launchtime = starttime
        comkey$ = "" ' Exit DO
    END IF
END IF

LOOP
  delay = FMdetect.bubble(launchtime, graph)

  IF delay = -1 THEN
    IF graph = FALSE THEN
      VIEW PRINT 4 TO 13
      LOCATE 13
      PRINT "No bubble found."
      VIEW PRINT
    END IF
    PRINT #DATAFILENUM, "No bubble found."
  EXIT FOR
  END IF

END IF

CALL monitor.furnace(temperature)

delta.pressure = pressure - delta.pressure

IF bubbler = 1 THEN
    CALL bubbl.stats(delay, bubb1.count, bubb1.ave, bubb1.stddev, reset.flag)
ELSE
    CALL bubbl.stats(delay, bubb2.count, bubb2.ave, bubb2.stddev, reset.flag)
        delta.delay = bubb2.ave - bubb1.ave
        delta.stddev = SQRT(bubb1.stddev^2 + bubb2.stddev^2)
        percent = 100*delta.stddev / delta.delay
        ave.velocity = DEPTH.DIFF/delta.delay
        density = delta.pressure * 70.31 / DEPTH.DIFF
    END IF

    ' If not drawing graphs, print results to the screen
    IF graph = FALSE THEN
        VIEW PRINT 4 TO 13
        LOCATE 13
        IF bubbler = 1 THEN
            PRINT USING "## ## ": bubbler; bubb1.count;
            PRINT USING "## ## ": temperature;
            PRINT USING "## ## ": delay; bubb1.ave
        ELSE
            PRINT USING "## ## ": bubbler; bubb2.count;
            PRINT USING "## ## ": temperature;
            PRINT USING "## ## ": delay; bubb2.ave
        END IF
    END IF
PRINT USING "UUU ; bubbler; bubb1.count;
PRINT USING "UUUU.U ; temperature;
PRINT USING "UUUUUU ; delay; bubb1.ave;
PRINT USING "UUUUUU ; delta.delay; delta.stddev; percent; ave.velocity
END IF

VIEW PRINT 17 TO 24
LOCATE 24
PRINT

IF bubbler = 1 THEN
PRINT USING "UUU ; bubbler, bubb1.count,
PRINT USING "UUUU.U ; pressure;
ELSE
PRINT USING "UUU ; bubbler, bubb2.count,
PRINT USING "UUUU.U ; pressure, delta.pressure, density;
END IF

END IF

* Print results to the file
IF printline ) MAXPRINTLINE THEN
PRINT @DATAFILENUM, TIMES, DATES, VERSION
PRINT @DATAFILENUM, "Bubble Count Temp. 	 Tile 	 Ave 	 Delta 	 Std Dev 	 % 	 ave veloc
PRINT @DATAFILENUM, "Pressure Delta 	 Density"
PRINT @DATAFILENUM, " #  	 deg. C Delay Delay Delay of Delay cm/sec"
PRINT @DATAFILENUM, " PSI  Pressure g/cm63"
PRINT @DATAFILENUM, " ___________ "
printline = 4
END IF

IF bubbler = 1 THEN
PRINT @DATAFILENUM, USING "UUU ; bubbler; bubb1.count;
PRINT @DATAFILENUM, USING "UUUU.U ; temperature;
PRINT @DATAFILENUM, USING "UUUUUU ; delay; bubb1.ave;
PRINT @DATAFILENUM, SPACE*(9+4+5);
PRINT @DATAFILENUM, USING "UUUU.U ; pressure
ELSE
PRINT @DATAFILENUM, USING "UUU ; bubbler; bubb2.count;
PRINT @DATAFILENUM, USING "UUUU.U ; temperature;
PRINT @DATAFILENUM, USING "UUUUUU ; delay; bubb2.ave;
PRINT @DATAFILENUM, USING "UUUUUU ; delta.delay; delta.stddev; percent; ave.velocity;
PRINT @DATAFILENUM, SPACE*(5);
PRINT @DATAFILENUM, USING "UUUU.U ; pressure, delta.pressure, density
END IF

printline = printline + 1

* We have to wait until the detector is purged before we launch another
 bubble. If we're graphing the helium response, we just keep watching
 it for a few more seconds. If we're not watching the graph then we
 wait here.
IF graph = FALSE THEN start=TIMER: WHILE TIMER-start(PURGETIME: WEND
comakey$ = inakey$

IF comakey$ = "g" OR comakey$ = "G" THEN graph = TRUE: CALL set.screen(graph) ' Turn on graphing
IF comakey$ = "l" OR comakey$ = "L" THEN graph = FALSE: CALL set.screen(graph) ' Turn off graphing
IF comakey$ = "r" OR comakey$ = "R" THEN reset.flag = TRUE ' Reset running averages
IF comakey$ = "q" OR comakey$ = "Q" THEN EXIT FOR ' We're done

NEXT bubbler

' If running averages have been reset then turn off reset.flag and
' inform the user.
IF reset.flag = TRUE THEN
    reset.flag = FALSE
    VIEW PRINT 25 TO 25
    LOCATE 25, 45
    PRINT "Running averages reset."
    printline = printline + 1
END IF

LOOP UNTIL comakey$ = "q" OR comakey$ = "Q"

CALL init.cards
CHDIR "D:ROB"
END

SUB set.screen(graph) STATIC
    IF graph = FALSE THEN
        SCREEN 9
        CLS 0
        VIEW PRINT 1 TO 4
        PRINT "Bubble Count Temp. Time Ave Delay Delay Std Dev cm/sec Ave velocity"
        PRINT "deg. C Delay Dolly Delay of Delay cm/sec"
        VIEW PRINT 14 TO 17
        PRINT "Bubble Count Pressure Delta Density"
        PRINT "# Pressure g/cm^3"
        PRINT "__ ______ ______ ______ ______ ______ ______ ____________________________ 
        END IF
    END SUB

SUB command.line(graph,bubbler) STATIC
    IF graph = FALSE THEN
        VIEW PRINT 25 TO 25
        CLS 2
        LOCATE 25
        PRINT "G: graphics, R:_reset averages, Q: Quit";
    ELSE
        ' Initialize graphics screen
        SCREEN 9
        CLS 0
        VIEW PRINT 25 TO 25
        LOCATE 25
        PRINT "L: List Data, R: Reset averages, Q: Quit"
        LOCATE 25, 50
        IF bubbler = 1 THEN
PRINT "Upper Bubblers";
ELSE
   PRINT "Lower Bubblers";
END IF
END IF
END SUB
'CntrlLib.bas
10/4/89
All the subroutines used by the various bubble launching programs.
Routines include card initialization, bubble launching, pressure and
helium monitoring.

REM #INCLUDE: 'D:ROBASOURCERCONSTLIB.BAS'
REM #INCLUDE: 'D:ROBASOURCERPLTPACK.BAS'
REM #INCLUDE: 'D:ROBASOURCERSTATSLIB.BAS'
REM #INCLUDE: 'D:ROBASOURCERNUTILLIB.BAS'

CONST OPENTIME = .5 ' Time for downstream valve to remain open

SUB read.pressure(bubbler, pressure.value, time) STATIC
' Subroutine to read the pressure transducer associated with 'bubbler.'
' We convert the value recorded from the transducer's analog input channel
' into a real gauge pressure in PSI and return that value in 'pressure.'
' The time of the sample is returned as 'time.'
    DIM at(10)
    DIM b(10)
    DIM slope(2)
    DIM intercept(2)

    * These are the linear regression parameters from the calibration
    * of the pressure transducers. Also included in the intercept is a
    * correction for deviation from atmospheric pressure. See Glass lab book
    * P. 24
    slope(1) = .9801 ' Slope for transducer 01
    intercept(1) = -1.069 + .0579 ' Intercept for transducer 01
    slope(2) = .9692 ' Slope for transducer 02
    intercept(2) = -1.062 + .0699 ' Intercept for transducer 02

    c$ = 'h' + CHR$(0)
    at(0) = bubbler
    CALL aml(at(0), b(0), c$)
    time = TIMER
    CALL convert(b(0), transvalue)
    pressure.value = transvalue * slope(bubbler) + intercept(bubbler)
END SUB

SUB find.launch(found.time, hydrostatic, bubbler, graph, overlap) STATIC
' Continuously monitors the pressure transducer associated with the 'bubbler'.
' The launch of a bubbler is indicated by a rapid drop in pressure.
' Function returns the time at which that drop occurs.
' If no launch is found after 'MAXLAUNCHTIME', routine returns -1.
' If graph is TRUE, the pressure profile will be plotted on the screen.
' If graph and overlap are both TRUE then the pressure profiles from several
' launches will be plotted over one another.
    CONST MAXLAUNCHTIME = 25 ' Maximum time to detect bubble launch (secs)
    CONST LEVELOFF.TIME = 2 ' Maximum time for pressure curve to level off.
    CONST HYDRO.SAMPLETIME = 2 ' Amount of time we accumulate stats to find hydrostatic pressure
    CONST DROP.THRESHOLD = .02 ' Absolute decrease in pressure for bubble launch to be signaled
    CONST OVERLAP.NUM = 8 ' Number of profiles to overlap.

    STATIC overlap.count ' Number of profiles drawn on graph
grid$ = TRUE ' Draw a grid on the plot

' Initialize the parameters to be found to their "not found" values
hydrostatic = -1
found.time = -1

IF graph = TRUE THEN
  ' Set the first point to be plotted
  CALL read.pressure(bubbler, last.pressure, now)
  last.time = now

  ' Set the graph limits.
xmin = 0
xmax = MAXLAUNCHTIME
ymin = -0.02
  ' If overlap is FALSE ymax is based on the first pressure point read
  ' otherwise it is set to some arbitrary value.
  IF overlap () TRUE THEN ymax = last.pressure ELSE ymax = .85

  ' This will clear the screen whenever the overlap screen needs to be
  ' redrawn. Note that if overlap is FALSE, overlap.count is always 0
  ' and the screen is always cleared and the graph port reset.
  IF overlap.count = 0 THEN
    ' Sometimes ymax is less than ymin so let set.user.coords straighten it out
    CALL set.physical.coords(0, 0, 600, 150)
    CALL set.user.coords(xmin, ymin, xmax, ymax)
    ymax = ymax + .15

  ' Set up the graphics screen
  CALL set.physical.coords(0, 0, 600, 150)
  CALL set.user.coords(xmin, ymin, xmax, ymax)
  CLS
  CALL box("Time", "Pressure", grid$)
END IF

IF overlap = TRUE THEN
  IF overlap.count (= OVERLAP.NUM THEN overlap.count=overlap.count+1 ELSE overlap.count = 0
END IF

END IF

' Find the pressure drop which indicates bubble launch
reset.flag = TRUE
starttime = TIMER
DO
  ' Read channels until pressure drop has been found
  ' Read a pressure value and add that value to the statistic
  CALL read.pressure(bubbler, pressure.value, now)
  CALL pressure.stats(pressure.value, level.count, level.stddev, reset.flag)

  IF level.ave-pressure.value ) DROP.THRESHOLD THEN ' Check for drop
    found.time = now ' We found the bubble launch
    BEEP ' Sound off
    IF graph = TRUE THEN
      LINE (found.time-starttime,usr.ymin) - (found.time-starttime,usr.ymax), 5
    END IF
  END IF
END DO
EXIT DO

END IF

IF graph = TRUE THEN
   LINE (last.time-starttime,last.pressure) - (now-starttime, pressure.value), 3
   last.pressure = pressure.value
   last.time = now
END IF
reset.flag = FALSE
LOOP UNTIL ABS(TIME-starttime) > MAXLAUNCHTIME

' Now find the leveling off of the pressure curve.
' We consider it level if the slope of two consecutive pairs of points
' is less than some specified value.
IF found.time () -1 THEN
   last.time = found.time
   leveloff = FALSE
   DO
      last.pressure = pressure.value
      CALL read.pressure(bubbler, pressure.value, now)

      IF abs(pressure.value-last.pressure) (.05*DROP.THRESHOLD THEN ' Check for leveling off
         IF leveloff = TRUE THEN
            EXIT DO
         ELSE
            leveloff = TRUE
         END IF
      ELSE
         leveloff = FALSE
      END IF

      IF graph = TRUE THEN
         LINE (last.time-starttime,Isst.pressure) - (now-starttiwe,pressure.value), 3
         last.time = now
      END IF
   END DO

   IF graph = TRUE THEN
      LINE (last.time-starttime,last.pressure) - (now-starttime, pressure.value), 3
      last.time = now
   END IF
LOOP UNTIL ABS(TIME-found.time) > LEVELOFF.TIME

' Now find the hydrostatic pressure
reset.flag = TRUE
last.time = now
last.pressure = pressure.value
' Mark beginning of hydrostatic pressure average.
IF graph = TRUE THEN LINE (now-starttiiae, usr.yain) - (now-starttiiae, usr.ymax), 4
start.hydrotiie = TIMER
DO
   CALL read.pressure(bubbler,pressure.value, now)
   CALL pressure.stats(pressure.value, level.count, level.ave, level.stddev, reset.flag)
   IF graph = TRUE THEN
      LINE (last.time-starttime,last.pressure) - (now-starttime, pressure.value), 3
      last.pressure = pressure.value
      last.time = now
   END IF
   reset.flag = FALSE
LOOP UNTIL ABS(TIMER-start.hydrotime) > HYDRO.SAMPLETIME

hydrostatic = level.ave
' Command to one analog channel once
a%(0) = 3
CALL asl(a%(0),b(0),d)
time = TIMER
CALL convert(b(0),level)
level = -level

DEF FN detect.bubble(starttime,graph)
' Continuously read the helium level, looking for a rise in
' the helium level (analog channel 3). Stop the timer when rise is found.
' If the bubble is not found after a time MAXDETECTTIME the loop
' times out and 'delay' remains unchanged.
' Reset the Helium level running average before exiting subroutine.
CONST MAXDETECTTIME = 40  ' Maximum time to detect bubble (secs)
CONST THRESHOLD = .05    ' Absolute increase in He level for rise to be signaled

IF graph = TRUE THEN
   ' Set the graph limits
   xmin = 0
   xmax = MAXDETECTTIME
   ymin = 0.0
   ymax = 6.0
   grid% = TRUE  ' Draw a grid on the plot
   CALL set.physical.coords(0,170,600,330)
   CALL set.user.coords(xmin,ymin,xmax,ymax)
   CALL box("Time", "He. Level", grid%)
   ' Set the first point to be plotted
   CALL read.helium(last.level,now)
   last.time = now
END IF

reset.flag = TRUE
DO    ' Read channels until helium rise has been found
   ' Read a helium level and add that value to the statistics
   CALL read.helium(helium.level,now)
   CALL HeLevel.stats(helium.level, level.count, level.ave, level.stddev, reset.flag)
   IF helium.level-level.ave ) THRESHOLD THEN  ' Check for rise
      found.time = now
FNdetect.bubble = found.time - starttime
BEEP  ' Sound off
IF graph = TRUE THEN
    LINE (found.time-starttime,usr.ymin) - (found.time-starttime,usr.ymax), 5
END IF
EXIT DO  ' Exit read loop
END IF

IF graph = TRUE THEN
    LINE (last.time-starttime,last.level) - (now-starttime, helium.level), 3
    last.level = helium.level
    last.time = now
END IF

IF TIMER-starttime > MAXDETECTTIME THEN
    VIEW PRINT 25 TO 25
    LOCATE 25,1
    PRINT "Timeout after "; MAXDETECTTIME; " seconds. No bubble found."
    found.time = -1
    FNdetect.bubble = -1
    EXIT DO
END IF
reset.flag = FALSE
LOOP
    IF graph = TRUE AND found.time () -1 THEN
        last.level = helium.level
        last.time = found.time
        DO WHILE TIMER - found.time ( MAXDETECTTIME/2
            CALL read.helium(helium.level, now)
            LINE (last.time-starttime,last.level) - (now-starttime, helium.level), 3
            last.level = helium.level
            last.time = now
        LOOP
    END IF
END IF
END DEF

SUB make.bubble(bubbler, starttime) STATIC
    ' Subroutine governs the generation of a bubble by alternately opening and closing the upstream and downstream valves. The bubbler through which a bubble is launched is determined by the parameter 'bubbler.' The instant at which the downstream valve is opened is returned in the parameter 'starttime.'
    DIM at(10)
    DIM b(10)

    ' We select the bubbler into which we inject the pulse.
    c$ = "0" + CHRS(0)  ' Command to set digital output in bits
    at(2) = 2 - bubbler  ' If bubbler=1: a(2)=1; if bubbler=2: a(2)=0
    at(0) = CLOSED : at(1) = CLOSED
    CALL sm1(at(0),b(0),c$)  ' Issue the command
    CALL pause(OPENTIME)

    ' We turn the relay on for the amount of time specified in 'OPENTIME.'
    ' This generates the bubble by opening and closing the flow valves.
at(0) = CLOSED : at(1) = CLOSED

' Open and close upstream valve.

at(0) = OPENED ' Set digital value for I/O BO = 1
CALL ami(at(0),b(0),c$) ' Issue the command
begin=TIMER: WHILE TIMER-begin < OPENTIME: VEND 'Timing Loop
at(0) = CLOSED ' Set digital value for I/O BO = 1
CALL ami(at(0),b(0),c$) ' Issue the command

' Open the downstream valve
CALL pause(OPENTIME)
at(1) = OPENED ' Set digital value for I/O 10 = 1
CALL ami(at(0),b(0),c$) ' Issue the command

start=TIMER

END SUB

SUB purge.bubblers STATIC
' Subroutine opens both the upstream and the downstream valves for a user-specified amount of time to purge the bubbler lines of air.

CONST AFTERPURGETIME = 10 ' Time for He to clear out of detector after purge

DIM at(10)
DIM b(10)

INPUT "Do you want to purge the bubblers (y or n)? ", response$
responses = MID$(response$,1,1)
IF response$ () "y" AND response$ () "Y" THEN EXIT SUB ' No purging to be done

' Purge both bubblers of air.
' First we select the bubbler.
c$ = "0" + CHR$(0) ' Command to set digital output in bits
FOR bubbler = 1 TO 2
    at(2) = 2 - bubbler ' If bubbler=1: a(2)=1; if bubbler=2: a(2)=0
    at(0) = CLOSED : at(1) = CLOSED ' Start the timer
    CALL ami(at(0),b(0),c$) ' Issue the command
    CALL pause(OPENTIME)
NEXT bubbler

' Now wait an amount of time specified by 'AFTERPURGETIME'.
' This allows He leak detector to purge.
CALL pause(AFTERPURGETIME)

END SUB

SUB convert(before,after) STATIC
' The OMEGA cards' driver uses the Microsoft Binary Format for floating point numbers. QuickBasic uses IEEE format. This routine converts the
' value 'before' from MFB format to IEEE format and returns the results
' in 'after'.

ptr = varptr(before)
sign = 1 - (peek(ptr+2) AND 128)/64
want = ((peek(ptr+2) OR 128)+2616 + peek(ptr+1)*268 + peek(ptr)) / 2623
expon = peek(ptr+3)-129
IF expon < -126 THEN expon = -126
IF expon > 127 THEN expon = 127
after = sign * want * 2**expon

SUB init_cards STATIC
' Subroutine initializes and calibrates the analog input and output channels
' and the digital output channels

DIM a%(10)
DIM b(10)

inchannels% = 5
range% = 5 ' +/- 5 Volts
thermotype% = 42 ' K type thermocouple (25 mV range)
delay% = 300 ' 100 * .03 milliseconds between analog channels

' Set the number of analog input channels
cs = "W" + CHR$(0)
a%(0) = inchannels%
CALL ani(a%(0),b(0),cs)

' Set the range for the analog input channels
cs = "r" + CHR$(0)
a%(0) = range%
a%(1) = range%
a%(2) = range%
a%(3) = range%
a%(4) = thermotype%
CALL ani(a%(0),b(0),cs)

' Set the delay for the analog input channels
cs = "d" + CHR$(0)
a%(0) = delay%
CALL ani(a%(0),b(0),cs)

' Calibrate the analog input channels AFTER setting their other parameters
cs = "c" + CHR$(0)
CALL ani(a%(0),b(0),cs)

' Set digital I/O #0-2 to output
cs = "S" + CHR$(0)
a%(0) = 1
a%(1) = 1
a%(2) = 1
a%(3) = 1
CALL ani(a%(0),b(0),cs)
' Set their status to high
cs = "0" + CHR$(0)
cl(0) = 1
e(1) = 1
e(2) = 1
e(3) = 1
CALL ani(at(0), b(0), cs)
END SUB
' INTEGRAT
' Program to launch bubbles and detect their presence with the He detector.
' Makes no attempt to nail down the launch time by monitoring the pressure
' profiles. The bubble size, as measured by the integration of the He detector
' signal, is displayed.

REM * INCLUDE: 'D:\QB\SOURCE\CNTRLIB.BAS'

CONST CLEARTIME = 7.0
CONST OUTPUTFILENUM = 1

graph = TRUE
overlap = TRUE
react.flag = FALSE
bubler = 2

' Open output file
INPUT "Enter name for output file: ", outfile$  
HOUR "D:\QB\DATA"
OPEN outfile$ FOR OUTPUT AS #OUTPUTFILENUM

' CALL purge(bubler)

SCREEN 2
VIEW  PRINT 1 TO 2
LOCATE 2
PRINT * "Size  Delay"
PRINT * ""
PRINT #OUTPUTFILENUM, * "Size  Delay"
PRINT #OUTPUTFILENUM, *

' Initialize the data acquisition and output and digital i/o cards
CALL init.cards

DO
    CALL make.bubble(bubler,launch.time)
    CALL detect.bubble(launch.time, detect.time, size, graph, overlap)
    delay = detect.time - launch.time
    VIEW PRINT 3 TO 12
    LOCATE 12
    PRINT USING " .8f .8f "; size; delay;
    PRINT
    PRINT #OUTPUTFILENUM, USING " .8f .8f "; size; delay;
    PRINT #OUTPUTFILENUM,

    CALL pause(CLEARTIME)
    comkey$ = FNucase$(INKEY$)
LOOP UNTIL comkey$ = '9'

CALL init.cards
CHDIR "D:\QB\SOURCE"
END
MONTEO
Program to execute launch bubbles and display pressure profiles to detect the launch. Makes no attempt to monitor the helium detector.

REM *INCLUDE: "D:\AB\SOURCE\CTRLIB.BAS"

CONST CLEARTIME = 7.0

graph = TRUE
overlap = FALSE

' Initialize the data acquisition and output and digital i/o cards
CALL init.cards

FOR bubbler = 1 TO 2
    CALL make.bubble(bubbler, start.time)
    CALL find.launch(bubbler, found.time, hydrostatic, overpressure, graph, overlap, times(), pressure(), count)

    comkey$ = INKEY$
    IF FNucase$(comkey$) = "G" THEN EXIT FOR
    CALL pause(CLEARTIME)
    NEXT bubbler

LOOP UNTIL FNucase$(comkey$) = "G"

CALL init.cards
ENDIS "D:\AB\SOURCE"
END
CALIB_PT

CALIB_PT

11/20/89

Program to facilitate calibration of the pressure transducers

Write a file which includes an average reading of the transducers

from the A/D card as well as a pressure reading from the precision

pressure gauge.

REM $INCLUDE: 'D:\OB\SOURCE\CNTALLIB.BAS'

CONST DATAFILENUM = 1
CONST WAITTIME = 1.

' Open data file
INPUT "Enter data file name: ", datafile$  
DATAFILE$ FOR OUTPUT AS DATAFILENUM
PRINT DATAFILENUM, "Gauge PT #1 PT #2"

' Initialize the data acquisition and output and digital i/o cards
CALL init.cards

count = 0

INPUT "Set the pressure to desired level (hit <CR> when ready, 'q' to quit) ", com$  
IF com$ = "q" OR com$ = "0" THEN EXIT DO

' Now we alternate opening the launchers to allow the pressure to equilibrate.
FOR i = 1 TO 3
   FOR bubbler = 1 TO 2
      CALL choose(bubbler)  
      CALL pause(WAITTIME)
   NEXT
   count = count + 1
   INPUT "Enter pressure from gauge: ", act.pressure
   PRINT DATAFILENUM, USING "8.##"; act.pressure,
   PRINT USING "8.##"; act.pressure,

FOR bubbler = 1 TO 2
   CALL choose(bubbler)
   reset.flag = TRUE
   FOR i = 1 to 50
      CALL read.pressure(bubbler, pressure, time)  
      CALL read.rawpressure(bubbler, pressure, time)  
      CALL rawpressure.stats(pressure, count, ave, stddev, reset.flag)
      reset.flag = FALSE
   NEXT
   PRINT DATAFILENUM, USING "8.0000"; ave,
   PRINT USING "8.##"; ave,
NEXT bubbler
PRINT DATAFILENUM,
PRINT
PRINT

LOOP

CALL init.cards
CHEIR "Dr\QB"
ENV

SUB choose(bubbler) STATIC
"Subroutine switches between the upper and lower launch tubes.
DIM aZ(10)
DIM b(10)

' Select the bubbler.
$: = "0" + CHR$(0)
aZ(2) = 2 - bubbler
If bubbler=1: a(2)=1; if bubbler=2: a(2)=0
aZ(0) = OPENED : aZ(1) = OPENED
CALL acl(aZ(0),b(0),c$) ' Issue the command

END SUB

SUB rawpressure(bubbler, pressure.value, time) STATIC
"Subroutine to read the pressure transducer associated with 'bubbler.'
The time of the sample is returned as 'time.'
DIM aZ(10)
DIM b(10)

c$ = "h" + CHR$(0)
aZ(0) = bubbler
CALL acl(aZ(0),b(0),c$)
time = TIMER
cALL convert(b(0),transvalue)
pressure.value = transvalue

END SUB

SUB raw-pressure.stats(newvalue, count, ave, stddev, reset.flag) STATIC
"Subroutine computes the new running average and standard deviation of the data
in 'buffer()' given the new data 'newvalue.'
To save time, the sum of the points is not calculated each time
the routine is called, rather the point which is falling out of the
running average is subtracted from the sums and the newest value is
added to the sums.

DIM buffer(1000)
STATIC thispoint, sum#, sumsquares#, buffer, localcount
CONST runavelength = 1000

' Reset all the values of the running average if the reset.flag has been
set to TRUE.
IF reset.flag = TRUE THEN
thispoint = 0
sum# = 0.0
sumsquares# = 0.0
localcount = 0
count = 0
ave = 0.0
stddev = 0.0
ERASE buffer

END IF

toss = buffer(thispoint)  ' Point falling out of the running average
buffer(thispoint) = newValue  ' Point coming in to the running average

' Before the buffer has filled, we have to keep track of how many points
' we have added to our running average.
IF localcount < runavelength THEN localcount = localcount + 1
count = localcount

' Recalculate the sum and sum of squares
sum# = sum# - toss + newValue
sumsquares# = sumsquares# - toss^2 + newValue^2

' Recalculate the average and standard deviation
ave = sum# / count
radicand = count * sumsquares# - sum#^2
IF count > 1 AND radicand >= 0 THEN stddev = SQR((radicand) / (count * (count - 1)))

IF thispoint = runavelength - 1 THEN
    thispoint = 0  ' The position for new points in the
ELSE
    thispoint = thispoint + 1  ' buffer advances from 0 to runavelength-1
END IF
FIT_LINE.BAS

```
' FIT_LINE
' Routine reads in the pressure transducer calibration file and calculates
' a linear regression for each transducer.

DIM act.pressure(50), trans.1(50), trans.2(50)

CONST DATAFILENUM = 1

' Open data file
INPUT "Enter data file name: ", datafile$
OPEN datafile$ FOR INPUT AS DATAFILENUM

' Read the heading line
LINE INPUT DATAFILENUM, heading$

COUNT = 0
DO WHILE NOT EOF(DATAFILENUM)
   COUNT = COUNT + 1
   INPUT DATAFILENUM, act.pressure(COUNT), trans.1(COUNT), trans.2(COUNT)
   PRINT act.pressure(COUNT), trans.1(COUNT), trans.2(COUNT)
   LINFIT(trans.1(), act.pressure(), COUNT, SLOPE, INTERCEPT, CORR)
   PRINT "Pressure(PSI) = ", USING 1 1.61116 1.61116; SLOPE;
   PRINT "Transducer1 + "; USING 1 1.61116 1.61116; INTERCEPT;
   PRINT "Correlation: "; USING 1 1.61116 1.61116; CORR
   PRINT
   LINFIT(trans.2(), act.pressure(), COUNT, SLOPE, INTERCEPT, CORR)
   PRINT "Pressure(PSI) = ", USING 1 1.61116 1.61116; SLOPE;
   PRINT "Transducer2 + "; USING 1 1.61116 1.61116; INTERCEPT;
   PRINT "Correlation: "; USING 1 1.61116 1.61116; CORR
   PRINT
   CHOR "D:108"
END

SUB linfit(x(), y(), COUNT, SLOPE, INTERCEPT, CORR) STATIC
' Subroutine takes as input a two dimensional array of
' and data. It performs a least squares fit on this series and returns
' the slope and intercept of the line.

SUMX = 0.
SUMXSQ = 0.
SUMY = 0.
SUMYSQ = 0.
SUMBOTH = 0.
```

Find the least-squares sums -
FOR \(i = 1\) TO \(\text{countX}\)
\[
\begin{align*}
\text{sumx} & = \text{sumx} + x(i) \\
\text{sumxsq} & = \text{sumxsq} + x(i)^2 \\
\text{sumy} & = \text{sumy} + y(i) \\
\text{sumysq} & = \text{sumysq} + y(i)^2 \\
\text{sumboth} & = \text{sumboth} + x(i)y(i)
\end{align*}
\]
NEXT \(i\)

\[
slope = \frac{(\text{countX}\text{sumboth} - \text{sumx}\text{sumy})}{(\text{countX}\text{sumxsq} - \text{sumx}^2)}
\]
\[
\text{intercept} = \frac{\text{sumy} - \text{slope}\text{sumx}}{\text{countX}}
\]
\[
\text{corr} = \frac{(\text{countX}\text{sumboth} - \text{sumx}\text{sumy})}{\text{ SSR}(\text{countX}\text{sumxsq} - \text{sumx}^2) + (\text{countX}\text{sumysq} - \text{sumy}^2)}
\]
END SUB
F7:72 ,1; to plot pressure and helium detection profiles from files saved
actual bubble generation runs.

REM $INCLUDE: 'D:\OB\SOURCE\PLOTPACK.BAS'
REM $INCLUDE: 'D:\OB\SOURCE\CONSTLIB.BAS'
REM $INCLUDE: 'D:\OB\SOURCE\UTILLIB.BAS'

COMET DATAFILENUM = 1
CONST MAXLAUNCHTIME = 25
CONST DELETEDTIME = 40

grid$ = TRUE  " Draw a grid on the plot

Loop through as many files as the user wants to plot
DO

Open data file
VIEW PRINT
INPUT "Enter data file name: ", datafile$  " Check for a line marking one of the special events
IF FNucase$(datafile$) = "B" THEN EXIT DO
CHDIR "D:\OB\DATA"
CLOSE DATAFILENUM
OPEN datafile$ FOR INPUT AS DATAFILENUM
LINE INPUT DATAFILENUM, comment$

' Set the first point
INPUT DATAFILENUM, last.time, last.pressure

' Set the graph limits.
xmin = 0
xmax = MAXLAUNCHTIME
ymin = 0
ymax = last.pressure

' Sometimes ymax is less than ymin so let set.user.coords straighten it out
CALL set.physical.coords(0,0,600,150)
CALL set.user.coords(xmin,ymin,xmax,ymax)
ymax = ymax + .15

' Set up the graphics screen
CALL set.physical.coords(0,0,600,150)
CALL set.user.coords(xmin,ymin,xmax,ymax)
CLS
CALL box("Time", "Pressure", grid$)

VIEW PRINT 25 TO 25
LOCATE 25
PRINT comment$;

' Plot the pressure points
DO WHILE NOT EOF(DATAFILENUM)
   ' Set the next point
   INPUT DATAFILENUM, time, pressure
   ' Check for a line marking one of the special events
IF time <= -100 THEN
SELECT CASE time
CASE IS = -100   ' Found overpressure average
  INPUT #DATAFILENUM, overtime, overpressure
  LINE (usr.xmin, overpressure) - (overtime, overpressure), 10
CASE IS = -200   ' Found launch time
  INPUT #DATAFILENUM, launch.time, toss
  LINE (launch.time, usr.ymin) - (launch.time, usr.ymax), 13
CASE IS = -300   ' Found hydrostatic average
  INPUT #DATAFILENUM, hydrotime, hydrostatic
  LINE (hydrotime, hydrostatic) - (usr.xmax, hydrostatic), 10
CASE ELSE
  EXIT DO
END SELECT
ELSE
  ' No special event so just plot the data
  LINE (last.time, last.pressure) - (time, pressure), 11
  last.time = time
  last.pressure = pressure
END IF

' Now plot the Helium detection profile

' Set the graph limits
xmin = 0
xmax = MAXDETECTTIME
ymin = 0.0
ymax = 6.0

' Set up the graphics screen
CALL set.physical.coords(0,170,600,330)
CALL set.user.coords(xmin,ymin,xmax,ymax)
CLS
CALL box("Time", "He. Level", grid2)

' Get the first Helium point
INPUT #DATAFILENUM, last.time, last.level

' Plot the Helium points
DO WHILE NOT EOF(DATAFILENUM)
  ' Get the next point
  INPUT #DATAFILENUM, time, he.level
  ' Check for a line marking some special event
  IF time <= -100 THEN
    SELECT CASE time
    CASE IS = -400   ' Found bubble detection time
      INPUT #DATAFILENUM, detect.time, toss
      LINE (detect.time, usr.ymin) - (detect.time, usr.ymax), 13
    CASE IS = -500   ' Found end of integration time
      INPUT #DATAFILENUM, end.time, toss
    END SELECT
  END IF

  ' Plot the Helium points
  LINE (time, he.level) - last.time, last.level
  last.time = time
  last.level = he.level
END WHILE

' End of the data processing loop
LINE (end.time, usr.ymin) - (end.time, usr.ymax), 13
CASE ELSE
    EXIT DO
END SELECT
ELSE
    ' No special event so just plot the data
    LINE (last.time, last.level) - (time, he.level), 11
    last.time = time
    last.level = he.level
ENDIF
LOOP
END
Plotpack
Subroutine package to perform basic plotting routines
8/25/89

COMMON SHARED scr.xmin%, scr.ymin%, scr.xmax%, scr.ymax%
COMMON SHARED txt.min%, txt.max%
COMMON SHARED usr.xmin, usr.ymin, usr.xmax, usr.ymax

SUB set.physical.coords(xmin%, ymin%, xmax%, ymax%) STATIC
    ! Set the physical limits of screen's graphics window. Makes sure that the max's are greater than the min's
    IF xmax% < xmin% THEN
        temp% = xmin%
        xmin% = xmax%
        xmax% = temp%
    END IF

    IF ymax% < ymin% THEN
        temp% = ymin%
        ymin% = ymax%
        ymax% = temp%
    END IF

    scr.xmin% = xmin%
    scr.xmax% = xmax%
    scr.ymin% = ymin%
    scr.ymax% = ymax%
    SCREEN 7 ' Set the display type
    VIEW (scr.xmin%, scr.ymin%) - (scr.xmax%, scr.ymax%) ' Set the viewport
END SUB

SUB set.txt(min%, max%) STATIC
    ! Set the rows into which text will be written. Makes sure that the max's are greater than the min's
    IF max% < min% THEN
        temp% = min%
        min% = max%
        max% = temp%
    END IF

    txt.min% = min%
    txt.max% = max%
    VIEW PRINT txt.min% TO txt.max% ' Set the text port
END SUB

SUB set.user.coords(xmin, ymin, xmax, ymax) STATIC
    ! Set the user coordinates of the graphics window that is defined by 'set.physical.coords.' Makes sure that the max's are greater than the min's
    IF xmax < xmin THEN
        temp = xmin
        xmin = xmax
        xmax = temp
    END IF

    IF ymax < ymin THEN

    END IF
temp = ymin
ymin = ymax
ymax = temp

END IF

usr.xmin = xmin
usr.ymin = ymin
usr.xmax = xmax
usr.ymax = ymax

WINDOW (usr.xmin, usr.ymin)-(usr.xmax, usr.ymax) ' Set the viewport limits

USR box(xlabel$, ylabel$, grid%) STATIC
' Draw a frame for a plot. We want to make sure we have enough room for labels, so we shrink the screen boundaries from where they have been previously set.

chars% = 15
lines% = 2
xmargin% = chars%*(640/79) ' Border on sides (I allow space for 'chars' characters)
ymargin% = lines%*(350/24) ' Border on top and bottom (I allow space for 'lines' lines)

' First decide where the x and y labels should go.
xlabelx% = ((scr.xmin + .5*(scr.xmax-scr.xmin)) * 79/640) + 1
ylabelx% = ((scr.xmax - 0.02*(scr.ymax-scr.xmin)) * 24/350) + 1
ylabel% = ((scr.xmin + .5*(scr.xmax-scr.xmin)) * 79/640) + 1
yratio% = (150/350)

' Now we shrink the plotting window.
scr.xmin% = scr.xmin + xmargin%
scr.xmax% = scr.xmax - ymargin%

CALL set.physical.coords(scr.xmin%, scr.ymin%, scr.xmax%, scr.ymax%)

' Put on the tick marks
VIEW PRINT
CALL u.ticks(grid%)
CALL yticks(grid%)

' Draw a box around the plotting region.
LINE (usr.xmin,usr.ymin) - (usr.xmax,usr.ymax), 15, B

' Put the x and y labels on the plot.
LOCATE ylabel, xlabel
PRINT xlabel

LOCATE ylabel, xlabel
PRINT ylabel

END SUB

USR ticks(grid%) STATIC
' Draw and label ticks on the x-axis. If grid$ is TRUE then make the tick marks into a grid

exponent% = int(log(usr.xmax-usr.xmin)/log(10)) - 1
mantis\% = int((usr.xmax-usr.xmin)/10^exponent\%) \ 'number between 10 and 100

SELECT CASE mantissa\%
  CASE IS < 20
    interval = 2.5 * 10^exponent\%
  CASE IS < 45
    interval = 5 * 10^exponent\%
  CASE IS < 90
    interval = 10 * 10^exponent\%
  CASE ELSE
    interval = 20 * 10^exponent\%
END SELECT

usr.xmin = INT(usr.xmin/interval) * interval
usr.xmax = CINT(usr.xmax/interval + .499999) * interval
CALL set.user.coords(usr.xmin,usr.ymin,usr.xmax,usr.ymax)

xticky\% = (scr.ymax \ 24/350) + 2
FOR tick = usr.xmin TO usr.xmax STEP interval
  xtickx\% = ((scr.xmin\% + ((tick-usr.xmin)/(usr.xmax-usr.xmin))(scr.xmax\%-scr.xmin\%)) \ 79/640) - 1
  IF grid\% = 1 THEN
    LINE (tick,usr.ymin) - (tick,usr.ymax)
  ELSE
    length = .03*(usr.ymax-usr.ymin)
    LINE (tick,usr.ymin) - (tick,usr.ymin+length)
  END IF
END FOR

LOCATE xtickx\%,xticky\%
SELECT CASE ABS(tick)
  CASE IS < 1
    PRINT USING ".###"; tick
  CASE IS < 10
    PRINT USING ".####"; tick
  CASE IS < 100
    PRINT USING ".#####"; tick
  CASE IS < 1000
    PRINT USING ".######"; tick
  CASE ELSE
    PRINT USING ".#####"; tick
END SELECT
NEXT tick

END SUB

EJE yticks(grid\%) STATIC
  ' Draw and label ticks on the y-axis. If grid\% is TRUE then make the tick
  ' marks into a grid

exponent\% = int(log(usr.ymax-usr.ymin)/log(10)) - 1
mantis\% = int((usr.ymax-usr.ymin)/10^exponent\%) \ 'number between 10 and 100

SELECT CASE mantissa\%
  CASE IS < 15
    interval = 2.5 * 10^exponent\%
CASE IS < 25
    interval = 5 * 10^exponent%
CASE IS < 50
    interval = 10 * 10^exponent%
CASE ELSE
    interval = 20 * 10^exponent%
END SELECT

usr.ymin = INT(usr.ymin/interval) * interval
usr.ymax = CINT(usr.ymax/interval + .499999) * interval
CALL set.user.coords(usr.xmin,usr.ymin,usr.xmax,usr.ymax)

ytickx% = (scr.xmax% 79/640) — 5
FOR tick = usr.ymin TO usr.ymax STEP interval
    yticky% = ((scr.ymax% - ((tick — usr.ymin)/(usr.ymax—usr.ymin))*(scr.ymax%—scr.ymin%)) 24/350) + 1

    ' Put the y ticks on the plot.
    IF grid2 = 1 THEN
        LINE (usr.xmin,tick) — (usr.xmax,tick)
    ELSE
        length = .03*(usr.xmax—usr.xmin)
        LINE (usr.xmin,tick) — (usr.xmin+length,tick)
    END IF

LOCATE ytickx%, ytickx%
SELECT CASE ABS(tick)
    CASE IS < 1
        PRINT USING "4.444"; tick
    CASE IS < 10
        PRINT USING "44.44"; tick
    CASE IS < 100
        PRINT USING "444.4"; tick
    CASE IS < 1000
        PRINT USING "4444"; tick
    CASE ELSE
        PRINT USING "44.4"; tick
END SELECT
NEXT tick
END SUB

SUB limits(xdata(1), ydata(1), numpoints%) STATIC
    ' Subroutine to set the limits of the plotting window given the data arrays.
    ' Sets these limits a little larger than the max and min of the arrays to
    ' make the plotted data easier to see.
    margin = .05    ' Space allowed around the x and y limits.

    ' Find the max and min for the x and y data arrays.
    xmin = xdata(1)
    xmax = xdata(1)
    ymin = ydata(1)
    ymax = ydata(1)

    FOR i = 2 TO numpoints%
        IF xdata(i) > xmax THEN
            xmax = xdata(i)
        IF ydata(i) > ymax THEN
            ymax = ydata(i)
        END IF
    END FOR
END SUB
ELSEIF xdata(i) < xmin THEN
    xmin = xdata(i)
END IF

IF ydata(i) > ymax THEN
    ymax = ydata(i)
ELSEIF ydata(i) < yuin THEN
    yuin = ydata(i)
END IF

NEXT i

' Add some space around the limits.
xmin = xmin - space*hmin
xmax = xmax + space*hmax
ymin = ymin - space*ymin
ymax = ymax + space*ymax

'RESET THE PLOTTING WINDOWS LIMITS
CALL set.user.coords(xmin, ymin, xmax, ymax)

' call printcommon("n")

END SUB

SUB plot(xdata(1), ydata(1), numpoints%) STATIC
    FOR i = 2 TO numpoints%
        LINE (xdata(i-1), ydata(i-1)) - (xdata(i), ydata(i))
    NEXT i
END SUB

SUB printcommon(printer$) STATIC
    ' Diagnostic tool to print out the contents of the common block.
    ' If printer$ = "Y" then the output will go to the printer not to the screen.
    line1$ = "scr.xmin% = " + str(scr.xmin%) + ", scr.xmax% = " + str(scr.xmax%)
    line2$ = "scr.ymin% = " + str(scr.ymin%) + ", scr.ymax% = " + str(scr.ymax%)
    line3$ = "box.xmin = " + str(box.xmin) + ", box.xmax = " + str(box.xmax)
    line4$ = "box.ymin = " + str(box.ymin) + ", box.ymax = " + str(box.ymax)
    line5$ = "usr.xmin = " + str(usr.xmin) + ", usr.xmax = " + str(usr.xmax)
    line6$ = "usr.ymin = " + str(usr.ymin) + ", usr.ymax = " + str(usr.ymax)
    line7$ = "txt.xmin% = " + str(txt.xmin%) + ", txt.xmax% = " + str(txt.xmax%)

    IF printer$ = "Y" OR printer$ = "y" THEN
        LPRINT line1$
        LPRINT line2$
        LPRINT line3$
        LPRINT line4$
        LPRINT line5$
        LPRINT line6$
        LPRINT line7$
    ELSE
        PRINT line1$
        PRINT line2$
        PRINT line3$
        PRINT line4$
        PRINT line5$
        PRINT line6$
        PRINT line7$
    END IF
END SUB
' Program to communicate with the Lindberg Furnace Controller. You have to
' know the correct communication mnemonics to get any useful information.
CONST TRUE = 1, FALSE = 0
CONST FURNACENUM = 1
CONST MAXTRYS = 5
CONST ADDRESS = "0000"
CONST SIX = &H2, ETX = &H3, EOT = &H4
CONST END = &H5, ACK = &H6, NAK = &H15

' Function calculates the checksum for a controller command string.
' The checksum character is sent as the last character if the EOT, ACK
' or NAK character is not sent.
DEF FNC checksum(packet$)
    sum = 0
    FOR x = 2 to LEN(packet$)
        sum = sum XOR ASC(MIDS(packet$,x,1))
    NEXT x
    FNC checksum = sum
END DEF

DEF FNUcase$(instring$)
    length = LEN(instring$)
    FOR i = 1 TO length
        ch = ASC(MIDS(instring$,i,1))
        IF ch > 96 AND ch < 127 THEN
            MIDS(instring$,i,1) = CHRS(ch-32)
        END IF
    NEXT i
    FNUcase$ = instring$
END DEF

cotileS = "COM2:9600,E,7,1"
OPEN cotileS AS #FURNACENUM

DO
    INPUT "Read(R) or Write(W) a parameter? ", responses$
    IF responses$ = "Q" OR responses$ = "q" THEN EXIT DO
    IF responses$ = "R" OR responses$ = "r" THEN CALL readparam
    IF responses$ = "W" OR responses$ = "w" THEN CALL writeparam
LOOP

SUB readparam STATIC
    INPUT "Read what parameter? ", parameter$
    parameters$ = FNUcase$(parameter$)
    packets$ = CHR$(EOT) + ADDRESS + parameters$ + CHR$(END)
    PRINT #FURNACENUM, packets$;
    trys = 0
    DO
        PRINT #FURNACENUM, packets$;
        CALL get.response(response$,out.status)
        trys = trys + 1
    LOOP
END
LOOP WHILE out.status = FALSE AND trys <= MAXTRYS

IF out.status = FALSE THEN
  PRINT "No Response."
ELSE
  PRINT responses; " = "; responses
END IF
PRINT
END SUB

SUB writeparam STATIC
  INPUT "Write what parameter? ", parameters
  INPUT "New value? ", values

  parameters = FNucase$(parameters)
  packet$ = CHR$(STX) + parameters + values + CHR$(ETX)
  sum$ = CHR$(FNCchecksum(packet$))
  packet$ = CHR$(EOT) + ADDRESS + packet$ + sum$

  trys = 0
  DO
    PRINT furnacenum, packets;
    CALL get.response(response$, out.status)
    trys = trys + 1
  LOOP WHILE out.status = FALSE AND trys <= MAXTRYS

  IF out.status = FALSE THEN
    PRINT "Not Accepted."
  ELSE
    PRINT "Accepted"
  END IF
  PRINT
END SUB

SUB get.response(response$, status) STATIC

  CONST TIMEOUTTIME = 1

  responses$ = ""
  endtrans = FALSE
  starttime = TIMER
  DO WHILE endtrans = FALSE AND (TIMER-starttime <= TIMEOUTTIME)
    DO WHILE NOT EOF(furnacenum)
      newchars = INPUT$(1, FURNACENUM)
      IF ASC(newchars) = ACK OR ASC(newchars) = NAK OR ASC(newchars) = ETX THEN
        endtrans = TRUE
        response$ = response$ + newchars
      END IF
    LOOP

    LOOP
      IF response$ = "" THEN checknum = -1 ELSE checknum = ASC(newchars)
      SELECT CASE checknum
      CASE IS = -1
        response$ = "NO RESPONSE"
        status = FALSE
      CASE IS = ETX
        stxpos = instr(response$, CHR$(STX))
IF stxpos = 0 THEN
  responses$ = "BAD RESPONSE"
  status = FALSE
ELSE
  etxpos = instr(response$,CHR$(ETX))
  responses$ = mid$(response$,stxpos+3,(etxpos-stxpos)-3)
  status = TRUE
END IF
CASE IS = ACK
  response$ = CHR$(ACK)
  status = TRUE
CASE IS = NAK
  response$ = CHR$(NAK)
  status = FALSE
CASE ELSE
  lenres = len(response$)
  check$ = mid$(response$,1,lenres-1)
  stxpos = instr(response$,CHR$(STX))
  IF checksum = FNchecksum(check$) AND stxpos <> 0 THEN 'Good communication
    endpos = len(responses$)
    responses$ = mid$(response$,stxpos+3,(endpos-stxpos)-4)
    status = TRUE
  ELSE ' Communication messed up somewhere
    response$ = "TIMEOUT"
    status = FALSE
END IF
END SELECT
END SUB
' Functions and subroutines used by furnace communication routines.

CONST FURNACENUM = 2  ' File number for furnace communication through COM2

' Function calculates the checksum for a controller command string.
' The checksum character is sent as the last character if the EOT, ACK
' or NAK character is not sent.
DEF FN:checksum(packet$)
  sum = 0
  FOR x = 2 to LEN(packet$)
    sum = sum XOR ASC(MIDS(packet$,x,1))
  NEXT x
  FN:checksum = sum
END DEF

SUB monitor.furnace(temperature) STATIC
  ' Subroutine to return the temperature of the Linberg furnace as read by the
  ' Furnace Controller.
  ' Subroutine also insures that the furnace temperature does not rise
  ' beyond its rating. First, we read the temperature. If the temperature is
  ' within 10% of the rated value, we signal an alarm. If the temperature is
  ' above the rated value we set the output power of the furnace to zero until
  ' the temperature is within the rated value. When temperature returns to
  ' within range. We return automatic control to the furnace controller.

  DIM a%(10)
  DIM b%(10)

  CONST MAXTRYS = 5
  CONST ADDRESS = "0000"

  CONST STX = &H2, ETX = &H3, EOT = &H4
  CONST ENQ = &H5, ACK = &H6, NAK = &H15

  STATIC first
  uppertemp = 1200

  IF first = FALSE THEN
    comfile$ = "COM2:9600,E,7,1"
    OPEN comfile$ AS #FURNACENUM
  END IF

  INPUT "Enter the temperature limit for the furnace: ", uppertemp
  first = TRUE

END IF

' Read thermocouple used for temperature control from furnace controller.
  packets$ = CHR$(EOT) + ADDRESS + "PV" + CHR$(ENQ)
  trys = 0
  DO
    PRINT #FURNACENUM, packets$;
    CALL get.response(response$, status)
    trys = trys + 1
  LOOP WHILE status = FALSE AND trys = MAXTRYS
  IF status = FALSE THEN
    VIEW PRINT 25 TO 25
LOCATE 25,1
PRINT "Can't communicate with furnace controller."
EXIT SUB
ELSE
  temperature = val(response$)
END IF

' Now read integrated thermocouple from OMEGA A/D card.
c$ = "h" + CHR$(0)
a%(0) = 5
CALL ami(a%(0),b(0),c$)
CALL convert(b(0),checktemp)

IF checktemp > uppertemp THEN
  PRINT "WARNING: Temperature in furnace is too high. Temp. ="; checktemp
  PRINT "I'm shutting it down until temperature is back in range."

' Put the furnace in manual mode and set the output power to zero.
' Keep issuing the command until it is accepted.
DO
  ' First set controller to manual
  packet$ = CHR$(STX) + "SW0020" + CHR$(ETX)
  sum$ = CHR$(FNCchecksum(packet$))
  packet$ = CHR$(EOT) + ADDRESS + packet$ + sum$
  trys = 0
  DO
    PRINT #FURNACENUM, packet$;
    CALL get.response(toss$, man.status)
    trys = trys + 1
    LOOP WHILE man.status = FALSE AND trys <= MAXTRYS
  IF man.status = FALSE THEN PRINT "Can't set furnace controller to manual."

  ' Now set output power to zero.
  packet$ = CHR$(STX) + "OP0.0" + CHR$(ETX)
  sum$ = CHR$(FNCchecksum(packet$))
  packet$ = CHR$(EOT) + ADDRESS + packet$ + sum$
  trys = 0
  DO
    PRINT #FURNACENUM, packet$;
    CALL get.response(toss$, out.status)
    trys = trys + 1
    LOOP WHILE out.status = FALSE AND trys <= MAXTRYS
  IF out.status = FALSE THEN PRINT "Can't change output of furnace controller."
  LOOP UNTIL man.status = TRUE AND out.status = TRUE

' Now read integrated thermocouple from OMEGA A/D card until temperature is
' within the temperature rating of the furnace.
 pausedtime = 5
DO
  CALL pause(pausetime)
  PRINT "Inner furnace is too hot."
  PRINT "Temperature is "; checktemp; " degrees Celsius."
c$ = "h" + CHR$(0)
a%(0) = 5
CALL aml(a(0),b(0),c$)
CALL convert(b(0),checktemp)
LOOP WHILE checktemp > upper temp

' Return the furnace to automatic control.
PRINT "Furnace temperature within range. Temp. = ": checktemp
PRINT "Returning controller to automatic mode."
packet$ = CHR$(STX) + "$\text{0020}$" + CHR$(ETX)
sum$ = CHR$(FNCsum(packet$))
packet$ = CHR$(ETX) + ADDRESS + packet$ + sum$
trys = 0
DO
PRINT $FURNACENUM, packet$;
CALL get.response(toss$,status)
trys = trys + 1
LOOP WHILE status = FALSE AND trys = MAXTRYS
IF status = FALSE THEN PRINT "Can't set furnace controller to automatic": EXIT SUB
END IF

SUB get.response(response$, status) STATIC

CONST STX = &H2, ETX = &H3, EOT = &H4
CONST ENQ = &H5, ACK = &H6, NAK = &H15
CONST TIMEOUTTIME = 1

response$ = ""
endtrans = FALSE
starttime = TIMER
DO WHILE endtrans = FALSE AND (TIMER-starttime < TIMEOUTTIME)
DO WHILE NOT EOF($FURNACENUM)
newchar$ = INPUTS(1,$FURNACENUM)
IF ASC(newchar$) = ACK OR ASC(newchar$) = NAK OR ASC(newchar$) = ETX THEN endtrans = TRUE
response$ = response$ + newchar$
END IF
END WHILE
END SUB
response$ = CHR$(NAK)
status = FALSE
CASE ELSE
lenres = len(response$)
check$ = mid$(response$,1,lenres-1)
stxpos = instr(response$,CHR$(STX))
IF checksum = FNchecksum(check$) AND stxpos = 0 THEN 'Good communication
   endpos = len(response$)
   response$ = mid$(response$,stxpos+3,(endpos-stxpos)-4)
   status = TRUE
ELSE ' Communication messed up somewhere
   response$ = "TIMEOUT"
   status = FALSE
END IF
END SELECT
END SUB
STATSLIB.BAS

Tuesday, December 5, 1989 1:50 pm

Page 1

' Statslib
' Library of routines to accumulate running statistics on various quantities.

SUB delay.stats(newvalue, count, ave, stddev, reset.flag) STATIC
' Subroutine computes the new running average and standard deviation of the data
' in 'buffer[]' given the new data 'newvalue.'
' To save time, the sum of the points is not calculated each time
' the routine is called, rather the point which is falling out of the
' running average is subtracted from the sums and the newest value is
' added to the sums.

DIM buffer(1000)
STATIC thispoint, sum#, sumquares#, buffer, localcount
CONST runavelength = 1000

' Reset all the values of the running average if the reset.flag has been
' set to TRUE.
IF reset.flag = TRUE THEN
    thispoint = 0
    sum# = 0.0
    sumquares# = 0.0
    localcount = 0
    count = 0
    ave = 0.0
    stddev = 0.0
    ERASE buffer
END IF

toss = buffer(thispoint)  ' Point falling out of the running average
buffer(thispoint) = newvalue  ' Point coming in to the running average

' Before the buffer has filled, we have to keep track of how many points
' we have added to our running average.
IF localcount < runavelength THEN localcount = localcount + 1
    count = localcount

' Recalculate the sum and sum of squares
    sum# = sum# - toss + newvalue
    sumquares# = sumquares# - toss^2 + newvalue^2

' Recalculate the average and standard deviation
    ave = sum#/count
    radicand = count*sumquares# - sum#^2
    IF count > 1 AND radicand >=0 THEN stddev = SQR((radicand) / (count*(count-1)))

    IF thispoint = runavelength-1 THEN
        thispoint = 0  ' The position for new points in the
    ELSE
        thispoint = thispoint + 1  ' buffer advances from 0 to runavelength-1
    END IF

END SUB

SUB bubbl.stats(newvalue, count, ave, stddev, reset.flag) STATIC
' Subroutine computes the new running average and standard deviation of the data
' in 'buffer[]' given the new data 'newvalue.'
' To save time, the sum of the points is not calculated each time
' the routine is called, rather the point which is falling out of the
' running average is subtracted from the sums and the newest value is
' added to the sums.

DIM buffer(1000)
STATIC thispoint, sum#, ssquares#, buffer, localcount
CONST runavelength = 1000

' Reset all the values of the running average if the reset.flag has been
' set to TRUE.
IF reset.flag = TRUE THEN
    thispoint = 0
    sum# = 0.0
    ssquares# = 0.0
    localcount = 0
    count = 0
    ave = 0.0
    stddev = 0.0
    ERASE buffer
END IF

toss = buffer(thispoint)   ' Point falling out of the running average
buffer(thispoint) = newvalue ' Point coming in to the running average

' Before the buffer has filled, we have to keep track of how many points
' we have added to our running average.
IF localcount < runavelength THEN localcount = localcount + 1
    count = localcount

' Recalculate the sum and sum of squares
    sum# = sum# - toss + newvalue
    ssquares# = ssquares# - toss^2 + newvalue^2

' Recalculate the average and standard deviation
    ave = sum#/count
    radicand = count*ssquares# - sum#^2
    IF count > 1 AND radicand >= 0 THEN stddev = SQR((radicand) / (count*(count-1)))

IF thispoint = runavelength-1 THEN
    thispoint = 0       ' The position for new points in the
ELSE
    thispoint = thispoint + 1  ' buffer advances from 0 to runavelength-1 then wraps back to 0.
END IF

END SUB

SUB bubb2.stats(newvalue, count, ave, stddev, reset.flag) STATIC
' Subroutine computes the new running average and standard deviation of the data
' in 'buffer() given the new data 'newvalue.'
' To save time, the sum of the points is not calculated each time
' the routine is called, rather the point which is falling out of the
' running average is subtracted from the sums and the newest value is
' added to the sums.

DIM buffer(1000)
STATIC thispoint, sum, suesquares, buffer, localcount
CONST runavelength = 1000

' Reset all the values of the running average if the reset.flag has been
' set to TRUE.
IF reset.flag = TRUE THEN
  thispoint = 0
  sum = 0.0
  suesquares = 0.0
  localcount = 0
  count = 0
  ave = 0.0
  stddev = 0.0
  ERASE buffer
END IF

toss = buffer(thispoint)  ' Point falling out of the running average
buffer(thispoint) = newvalue  ' Point coming in to the running average

' Before the buffer has filled, we have to keep track of how many points
' we have added to our running average.
IF localcount < runavelength THEN localcount = localcount + 1
  count = localcount

' Recalculate the sum and sum of squares
sum = sum - toss + newvalue
suesquares = suesquares - toss^2 + newvalue^2

' Recalculate the average and standard deviation
ave = sum/count
radicand = count*suesquares - sum^2
IF count > 1 AND radicand >= 0 THEN stddev = SQRT((radicand) / (count*count-1))

IF thispoint = runavelength-1 THEN
  thispoint = 0  ' The position for new points in the
ELSE  ' buffer advances from 0 to runavelength-1
  thispoint = thispoint + 1  ' then wraps back to 0.
END IF

END SUB

SUB HeLevel.stats(newvalue, count, ave, stddev, reset.flag) STATIC
' Subroutine computes the new running average and standard deviation of the data
' in 'buffer()' given the new data 'newvalue.'
' To save time, the sum of the points is not calculated each time
' the routine is called, rather the point which is falling out of the
' running average is subtracted from the sums and the newest value is
' added to the sums.

DIM buffer(1000)
STATIC thispoint, sum, suesquares, buffer, localcount
CONST runalendarth = 1000

' Reset all the values of the running average if the reset.flag has been
' set to TRUE.
IF reset.flag = TRUE THEN
toss = buffer(thispoint) ' Point falling out of the running average
buffer(thispoint) = newvalue ' Point coming in to the running average

' Before the buffer has filled, we have to keep track of how many points
' we have added to our running average.
IF localcount < runavelength THEN localcount = localcount + 1

' Recalculate the sum and sum of squares
suset = suet - toss + newvalue
suesquarest = suesquarest - toss^2 + newvalue^2

' Recalculate the average and standard deviation
ave = sum# / count
radicand = count# * suesquares - sum#^2
IF count > 1 AND radicand > 0 THEN stddev = SQRT((radicand) / (count#(count-1)))

IF thispoint = runavelength - 1 THEN
   thispoint = 0 ' The position for new points in the
ELSE
   thispoint = thispoint + 1 ' buffer advances from 0 to runavelength-1
   then wraps back to 0.
END IF

END SUB

SUB pressure.stats(newvalue, count, ave, stddev, reset.flag) STATIC
' Subroutine computes the new running average and standard deviation of the data
' in 'buffer()' given the new data 'newvalue.'
' To save time, the sum of the points is not calculated each time
' the routine is called, rather the point which is falling out of the
' running average is subtracted from the sums and the newest value is
' added to the sums.

DIM buffer(1000)
STATIC thispoint, sum#, suesquares#, buffer, localcount
CONST runavelength = 1000

' Reset all the values of the running average if the reset.flag has been
' set to TRUE.
IF reset.flag = TRUE THEN
   thispoint = 0
   sum# = 0.0
   suesquares# = 0.0
   localcount = 0
   count = 0
   ave = 0.0

   END IF

   toss = buffer(thispoint) ' Point falling out of the running average
   buffer(thispoint) = newvalue ' Point coming in to the running average

   ' Before the buffer has filled, we have to keep track of how many points
   ' we have added to our running average.
   IF localcount < runavelength THEN localcount = localcount + 1

   ' Recalculate the sum and sum of squares
   suet = suet - toss + newvalue
   suesquarest = suesquarest - toss^2 + newvalue^2

   ' Recalculate the average and standard deviation
   ave = sum# / count
   radicand = count# * suesquares - sum#^2
   IF count > 1 AND radicand > 0 THEN stddev = SQRT((radicand) / (count#(count-1)))

   IF thispoint = runavelength - 1 THEN
      thispoint = 0 ' The position for new points in the
   ELSE
      thispoint = thispoint + 1 ' buffer advances from 0 to runavelength-1
      then wraps back to 0.
   END IF

   END SUB

SUB pressure.stats(newvalue, count, ave, stddev, reset.flag) STATIC
' Subroutine computes the new running average and standard deviation of the data
' in 'buffer()' given the new data 'newvalue.'
' To save time, the sum of the points is not calculated each time
' the routine is called, rather the point which is falling out of the
' running average is subtracted from the sums and the newest value is
' added to the sums.

DIM buffer(1000)
STATIC thispoint, sum#, suesquares#, buffer, localcount
CONST runavelength = 1000

' Reset all the values of the running average if the reset.flag has been
' set to TRUE.
IF reset.flag = TRUE THEN
   thispoint = 0
   sum# = 0.0
   suesquares# = 0.0
   localcount = 0
   count = 0
   ave = 0.0

   END IF

   toss = buffer(thispoint) ' Point falling out of the running average
   buffer(thispoint) = newvalue ' Point coming in to the running average

   ' Before the buffer has filled, we have to keep track of how many points
   ' we have added to our running average.
   IF localcount < runavelength THEN localcount = localcount + 1

   ' Recalculate the sum and sum of squares
   suet = suet - toss + newvalue
   suesquarest = suesquarest - toss^2 + newvalue^2

   ' Recalculate the average and standard deviation
   ave = sum# / count
   radicand = count# * suesquares - sum#^2
   IF count > 1 AND radicand > 0 THEN stddev = SQRT((radicand) / (count#(count-1)))

   IF thispoint = runavelength - 1 THEN
      thispoint = 0 ' The position for new points in the
   ELSE
      thispoint = thispoint + 1 ' buffer advances from 0 to runavelength-1
      then wraps back to 0.
   END IF

   END SUB
```basic
DIM buffer(1000)
STATIC thispoint, sum#, sumquares#, buffer, localcount
CONST runavelength = 1000

' Reset all the values of the running average if the reset.flag has been
' set to TRUE.
IF reset.flag = TRUE THEN
    thispoint = 0
    sum# = 0.0
    sumquares# = 0.0
    localcount = 0
    count = 0
    ave = 0.0
    stddev = 0.0
    ERASE buffer
END IF

toss = buffer(thispoint)    ' Point falling out of the running average
buffer(thispoint) = newvalue    ' Point coming in to the running average

' Before the buffer has filled, we have to keep track of how many points
' we have added to our running average.
IF localcount < runavelength THEN localcount = localcount + 1
    count = localcount

' Recalculate the sum and sum of squares
suet = suet - toss + newvalue
sumquares = sumquares - toss^2 + newvalue^2

' Recalculate the average and standard deviation
ave = sum# / count
radicand = count*sumquares - sum#^2
IF count > 1 AND radicand > 0 THEN stddev = SQR((radicand) / (count*(count-1)))

IF thispoint = runavelength-1 THEN
    thispoint = 0    ' The position for new points in the
ELSE
    thispoint = thispoint + 1    ' buffer advances from 0 to runavelength-1
        ' then wraps back to 0.
END IF
```

This subroutine computes the new running average and standard deviation of the data in the buffer, given the new data 'newvalue.' To save time, the sum of the points is not calculated each time the routine is called, rather the point which is falling out of the running average is subtracted from the sums and the newest value is added to the sums.
Before the buffer has filled, we have to keep track of how many points we have added to our running average.

IF localcount < runavelength THEN localcount = localcount + 1

count = localcount

Recalculate the sum and sum of squares
sum$ = sum$ - toss + newvalue
sumsquares$ = sumsquares$ - toss^2 + newvalue^2

Recalculate the average and standard deviation
ave = sum$ / count
radicand = count * sumsquares$ - sum^2
IF count > 1 AND radicand > 0 THEN stddev = SQR((radicand) / (count * (count - 1)))

IF thispoint = runavelength - 1 THEN
    thispoint = 0
    ' The position for new points in the buffer advances from 0 to runavelength-1
ELSE
    thispoint = thispoint + 1
    ' then wraps back to 0.
END IF

END SUB
‘ SELTEMP
‘ Program to scan data files and assemble velocity data which correspond
‘ to sufficiently narrow temperature bands.
‘ Then accumulates statistics for data of those temperature bands.

REM $INCLUDE: "D:\OB\SOURCE\CONSTLIB.BAS"
REM $INCLUDE: "D:\OB\SOURCE\STATSLIB.BAS"

CONST NAMESFILENUM = 1
CONST DATAFILENUM = 2
CONST BUBBFILENUM = 3
CONST BUBB2FILENUM = 4

‘ THRESHOLD sets the maximum acceptable deviation from the desired temperature
CONST THRESHOLD = .5

‘ MINDATA sets the minimum number of points which must be present at each
temperature before we keep the accumulation.
CONST MINDATA = 10

CONST DEPTH.DIFF = 4.29 ' Difference in depth of launchers (cm)

‘ Errors may occur when reading the file. If that happens just
‘ continue reading the next line.
ON ERROR GOTO HANDLER

‘ Also open an output files for the results from each bubbler
CHDIR "D:\OB\DATA"
OPEN "bubblacc.dat" FOR OUTPUT AS IBUBBFILENUM
OPEN "bubb2acc.dat" FOR OUTPUT AS IBUBB2FILENUM

‘ Loop through all the temperatures of interest
FOR selecttemp = 986 TO 1067

‘ Reset the accumulated statistics on the first pass
resetbubbl.flag = TRUE
resetbubb2.flag = TRUE
resettempep.flag = TRUE
bubbl.count = 0
bubb2.count = 0
temp.count = 0

‘ Open file containing names of data files
CLOSE(NAMESFILENUM) ‘ First close it in case it’s already open.
OPEN "ranges.dat" FOR INPUT AS #NAMESFILENUM

‘ Read in the names of each data file
DO WHILE NOT EOF(NAMESFILENUM)
   INPUT #NAMESFILENUM, datafilename$, mintemp, maxtemp

   ‘ Look at the temperature ranges of the file to see if we should look here.
   IF mintemp < selecttemp AND selecttemp < maxtemp THEN
      PRINT datafilename$, "Temperature: "; selecttemp

   ‘ Open data file
   CLOSE(DATAFILENUM) ‘ First close it in case it’s already open.
OPEN datafilename$ FOR INPUT AS #DATAFILENUM

READFILE:
  ' Loop through each line in the data file
  DO WHILE NOT EOF(DATAFILENUN)
    INPUT #DATAFILENUM, bubbler, count, temperature, delay ' Get the data
    LINE INPUT #DATAFILENUM, rest$ ' Read the rest of the line

    ' Check the temp to see if its close enough to the specified temperature.
    IF ABS(temperature-selecttemp) < THRESHOLD THEN
      IF bubbler = 1 THEN
        CALL bubb1.stats(delay, bubb1.count, bubb1.ave, bubb1.stddev, resetbubbl.flag)
        PRINT "Got bubbler 1. Count = "; bubb1.count

        ' Don't reset the accumulated statistics after the first pass
        resetbubbl.flag = FALSE
      ELSEIF bubbler = 2 THEN
        CALL bubb2.stats(delay, bubb2.count, bubb2.ave, bubb2.stddev, resetbubb2.flag)
        PRINT "Got bubbler 2. Count = "; bubb2.count

        ' Don't reset the accumulated statistics after the first pass
        resetbubb2.flag = FALSE
      END IF
    END IF
  LOOP
END IF

CALL temp.stats(temperature, temp.count, temp.ave, temp.stddev, resettemp.flag)
resettemp.flag = FALSE

END IF

END

' Program segment allows recovery from some errors.
HANDLER:
  errmsg = ERR
  IF errmsg = 62 THEN ' Input past end error
    PRINT "Input past end. Resuming with next file."
    RESUME READFILE
  ELSE
    ERROR errmsg
  END IF

NEXT selecttemp

CHDIR "D:\OB"

END
ON ERROR GOTO 0
END IF

SUB displayit(bubblerZ, temp.ave, bubb.ave, bubb.stddev, bubb.count) STATIC
' Subroutine to print data in a format which is easy for a person to read.
STATIC been.herel, been.here2

IF bubblerZ = 1 THEN
  IF been.herel <> TRUE THEN
    PRINT "Temperature Ave. Delay std dev count"
    PRINT
    PRINT #BUBB1FILENUM, "Temperature Ave. Delay std dev count"
    PRINT #BUBB1FILENUM, "Temperature Ave. Delay std dev count"
    been.herel = TRUE
  END IF
  PRINT USING "#111.1	"; temp.ave,
  PRINT USING "#41.###	"; bubb.ave, bubb.stddev,
  PRINT USING "##	"; bubb.count
  PRINT #BUBB1FILENUM, USING "####.#	"; temp.ave,
  PRINT #BUBB1FILENUM, USING "##.###	"; bubb.ave, bubb.stddev,
  PRINT #BUBB1FILENUM, USING "##	"; bubb.count
ELSEIF bubblerZ = 2 THEN
  IF been.here2 <> TRUE THEN
    PRINT "Temperature Ave. Delay std dev count"
    PRINT
    PRINT #BUBB2FILENUM, "Temperature Ave. Delay std dev count"
    PRINT #BUBB2FILENUM, "Temperature Ave. Delay std dev count"
    been.here2 = TRUE
  END IF
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
  PRINT USING "#####.0	"; temp.ave,
END IF
END SUB
BAROMETR.BAS

Wednesday, December 13, 1989 1:15 pm

Barometer

Given the barometric pressure as read by the lab barometer, the "local" and
"sea level" pressures will be calculated.

These values should reflect the location of the barometer.

CONST H = 304.8 ′?? Height above sea level (meters)
CONST phi = 33.75 ′ latitude

lat = phi * 3.1415926 / 180 ′ Convert latitude from degrees to radians.

INPUT "Observed barometer reading (mm Hg): ", Pr
INPUT "Temperature (Celsius): ", t

These values are determined by the system of units.

In this case, we find pressure in millimeters of mercury.

Pc = 760 ′ standard pressure at sea level (mm)/
L = 0.0000184 ′(m/m degrees C)
M = 0.0001818 ′(m^3/m^3 degrees C)
to = 0 ′ degrees C
ts = 0 ′ degrees C

Ct = Pr * ( (1+L*(t-ts) / (1+M*(t-ts))) - 1)
Pt = Ct + Pr

x1 = 980.616/980.665
x2 = 1-0.00263731*cos(2*lat)+0.00000591*cos(2*lat)^2
Cg = Pr * (x1*x2 - 1)
Pl = Cg + Pt

deltaPs = Po * (1 - (1 - (0.0065/288.16*M)^5.2561)
P = Pl + deltaPs

MetricP1 = Pl
MetricP = P

Now calculate the same thing in English units.

These values are determined by the system of units.

In this case, we find pressure in inches of mercury.

Po = 29.921 ′ standard pressure at sea level (mm)
L = 0.0000102 ′(in/in degrees F)
M = 0.0001010 ′(in^3/in^3 degrees F)
to = 32 ′ degrees F
ts = 62 ′ degrees F

Convert pressure to in. Hg
Pr = Pr * .03937

Convert temperature to degrees F

Pt = Ct + Pr

" Convert pressure to in. Hg
Pr = Pr * .03937

" Convert temperature to degrees F

Pt = Ct + Pr

x1 = 980.616/980.665
x2 = 1 - 0.00263731 \cos(2 \theta) + 0.00000591 \cos(2 \theta)^2 \\
Cg = Pr \times (x1 \times x2 - 1) \\
P1 = Cg + Pt \\

deltaPs = Po \times (1 - (1 - (0.0065/288.16)^4)H)^5.2561 \\
P = P1 + deltaPs \\

PRINT \\
PRINT "Local station pressure:", \\
PRINT USING "#0.00"; P1; \\
PRINT " in. Hg", \\
PRINT " or "; \\
PRINT USING "#0.000"; metricP1; \\
PRINT " mm Hg" \\
PRINT \\
PRINT "Sea level pressure:", \\
PRINT USING "#0.00"; P; \\
PRINT " in. Hg", \\
PRINT " or "; \\
PRINT USING "#0.000"; metricP; \\
PRINT " mm Hg" \\

END
DensStat

Accumulates density statistics for the entered data files

REM $INCLUDE: 'D:\DB\SOURCE\COMSTLIB.BAS'
REM $INCLUDE: 'D:\DB\SOURCE\STATSLIB.BAS'

CONST DATAFILENUM = 1
CONST DEPTH.DIFF = 5.2 ' Difference in depth of launchers (cm)
CONST RANGE = .5 ' Range around selected temperature for data to be acceptable
reset.flag = FALSE

' Open file containing names of data files
INPUT "Enter name of data file: ", datafile$
CHDIR "D:\DATA"
OPEN datafile$ FOR INPUT AS #DATAFILENUM

' INPUT "Input temperature: ", seltemp
DO WHILE NOT EOF(#DATAFILENUM)
  INPUT #DATAFILENUM, bubbler, bubbcount, temperature, delay, avedelay
  IF ABS(temperature - seltemp) (= RANGE THEN
    IF (bubbler = 1 OR bubbler = 2) AND temperature > 900 THEN
      IF bubbler = 1 THEN
        INPUT #DATAFILENUM, pressure
        LINE INPUT #DATAFILENUM, toss$
        CALL bubbl.stats(pressure, bubbl.count, bubbl.ave, bubbl.stddev, reset.flag)
      ELSE
        INPUT #DATAFILENUM, deltdelay, stddelay, pcntdelay, vel, pressure
        LINE INPUT #DATAFILENUM, toss$
        CALL bubb2.stats(pressure, bubb2.count, bubb2.ave, bubb2.stddev, reset.flag)
      END IF
    END IF
    CALL temp.stats(temperature, temp.count, temp.ave, temp.stddev, reset.flag)
    LINE INPUT $DATAFILENUM, toss$
  ELSE
    CALL temp.stats(temperature, temp.count, temp.ave, temp.stddev, reset.flag)
    LINE INPUT #DATAFILENUM, toss$
  END IF
LOOP

PRINT 'Bubble Count Ave Std Dev % ', datafile$
PRINT '
PRINT * Press of Press
PRINT '  
PRINT USING "#0.0000000;1; bubbl.count;";
PRINT USING "#0.0000000;1; bubbl.ave; bubbl.stddev; 100*bubbl.stddev/bubbl.ave"

PRINT USING "#0.0000000;2;ubb2.count;"
PRINT USING "#0.0000000;2;ubb2.ave;ubb2.stddev; 100*ubb2.stddev/ubb2.ave"
PRINT

PRINT "Temp Ave Std Dev % Density Std dev % 
PRINT "Count Temp of Temp g/cm^3 of Dens. 
PRINT '  
PRINT USING "#0.0000000;1; temp.count;"
PRINT USING "#0.0000000;1; temp.ave; temp.stddev; 100*temp.stddev/temp.ave;"
density = (bubb2.ave - bubbl.ave) / 70.31 / DEPTH.DIFF

dens.stddev = SQRT(bubbl.stddev^2 + bubb2.stddev^2)

PRINT USING "####"; density;
PRINT USING "####"; dens.stddev; 100*dens.stddev/density

LPRINT "Bubble Count Ave Std Dev % ", datafile$
LPRINT " # Press of Press 
LPRINT "
LPRINT USING "#### "; 1; bubbl.count;
LPRINT USING "######## "; bubbl.ave; bubbl.stddev; 100*bubbl.stddev/bubbl.ave

LPRINT USING "#### "; 2; bubb2.count;
LPRINT USING "######## "; bubb2.ave; bubb2.stddev; 100*bubb2.stddev/bubb2.ave
LPRINT
LPRINT "Temp Ave Std Dev % Density Std dev % 
LPRINT "Count Temp of Temp g/cm^3 of Dens. 
LPRINT "
LPRINT USING "#### "; temp.count;
LPRINT USING "######## "; temp.ave; temp.stddev; 100*temp.stddev/temp.ave

LPRINT USING "#### "; density;
LPRINT USING "######### "; dens.stddev; 100*dens.stddev/density
LPRINT:LPRINT
DelaStat
' Accumulates delay statistics for the entered data files

REM $INCLUDE: 'D:10B\SOURCE\CONSTLIB.BAS'
REM $INCLUDE: 'D:10B\SOURCE\STATSLIB.BAS'

CONST DATAFILENUM = 1
CONST DEPTH.DIFF = 5.2 ' Difference in depth of launchers (cm)
CONST RANGE = .5 ' Range around selected temperature for data to be acceptable
reset.flag = FALSE

' Open file containing names of data files
INPUT "Enter name of data file: ", datafile$
CHDIR "D:10B\DATA"
OPEN datafile$ FOR INPUT AS @DATAFILENUM

'INPUT "Input temperature: ", seltemp

DO WHILE NOT EOF(@DATAFILENUM)
   INPUT @DATAFILENUM, bubbler, bubbcnt, temperature, delay
   LINE INPUT @DATAFILENUM, toss$
   IF ABS(temperature - seltemp) <= RANGE THEN
      IF (bubbler = 1 OR bubbler = 2) AND temperature > 900 THEN
         IF bubbler = 1 THEN
            CALL bubb1.stats(delay, bubbl.count, bubbl.ave, bubbl.stddev, reset.flag)
         ELSE
            CALL bubb2.stats(delay, bubb2.count, bubb2.ave, bubb2.stddev, reset.flag)
         END IF
      END IF
      CALL temp.stats(temperature, temp.count, temp.ave, temp.stddev, reset.flag)
   END IF
   CALL temp.stats(temperature, temp.count, temp.ave, temp.stddev, reset.flag)
   PRINT
END IF

PRINT "Bubble Count Ave Std Dev % ", DATAFILE$
PRINT " # Delay Ave of Delay 
PRINT "_________________________"

PRINT USING "#### "; 1; bubbl.count;
PRINT USING "####.#### "; bubbl.ave; bubbl.stddev; 100*bubbl.stddev/bubbl.ave
PRINT USING "#### "; 2; bubb2.count;
PRINT USING "####.#### "; bubb2.ave; bubb2.stddev; 100*bubb2.stddev/bubb2.ave
PRINT
PRINT "Temp Ave Std Dev % Velocity Std dev % ",
PRINT "Count Temp of Temp cm/sec of Vel. 
PRINT "_________________________
PRINT USING "#### "; temp.count;
PRINT USING "####.#### "; temp.ave; temp.stddev; 100*temp.stddev/temp.ave;

velocity = DEPTH.DIFF / (bubb2.ave - bubbl.ave)
velstd = SGR(bubbl.stddev^2 + bubb2.stddev^2) % velocity / (bubb2.ave - bubbl.ave)
PRINT USING "####.#### "; velocity; velstd; 100*velstd/velocity
LPRINT "Bubble Count Ave Std Dev %", DATAFILE$
LPRINT " Delay of Delay"
LPRINT 

LPRINT USING "#"; 1; bubb1.count;
LPRINT USING "###.###"; bubb1.ave; bubb1.stddev; 100*bubb1.stddev/bubb1.ave

LPRINT USING "###"; 2; bubb2.count;
LPRINT USING "#####"; bubb2.ave; bubb2.stddev; 100*bubb2.stddev/bubb2.ave

LPRINT 

LPRINT "Temp Ave Std Dev % Velocity Std dev %"
LPRINT "Count Temp of Temp cm/sec of Vel."
LPRINT 
LPRINT USING "###"; temp.count;
LPRINT USING "#####"; temp.ave; temp.stddev; 100*temp.stddev/temp.ave;

LPRINT USING "###"; velocity; velstd; 100*velstd/velocity
LPRINT: LPRINT
' KEEPTEMP
' Program to scan data files and assemble velocity data which correspond
' to sufficiently narrow temperature bands. The entire data line from the
' original data file is written to a data file which is named according to
' the currently selected temperature.

REM $INCLUDE: 'D:\QB\SOURCE\CONSTLIB.BAS'

CONST NAMESFILENUM = 1
CONST DATAFILENUM = 2
CONST BU1MMNUM = 3
CONST BUBB2RANNUM = 4

' THRESHOLD sets the maximum acceptable deviation from the desired temperature
CONST THRESHOLD = .5

CONST SELECTBUB = 2

' These are the temperatures to be selected.
DATA 1001,1008,1011,1020
'tempcount' gives the number of items in the data statement
tempcount = 4

' Errors may occur when reading the file. If that happens just
' continue reading the next line, if possible
ON ERROR GOTO HANDLER

CHDIR "D:QB\DATA"

' Loop through all the temperatures of interest
FOR T = 1 TO tempcount
READ selecttemp

' Reset the accumulated statistics on the first pass
resetbub1.flag = TRUE
resetbub2.flag = TRUE

' Open file containing names of data files
CLOSE(NAMESFILENUM) ' First close it in case it's already open.
OPEN "ranges.dat" FOR INPUT AS NAMESFILENUM

' Read in the names of each data file
DO WHILE NOT EOF(NAMESFILENUM)
  INPUT NAMESFILENUM, datafilenamed, mintemp, maxtemp

  ' Look at the temperature ranges of the file to see if we should look here.
  IF mintemp < selecttemp AND selecttemp < maxtemp THEN
    PRINT datafilenamed, "Temperature: "; selecttemp

  ' Open data file
  CLOSE(DATAFILENUM) ' First close it in case it's already open.
  OPEN datafilenamed FOR INPUT AS DATAFILENUM

  READFILE:
    ' Loop through each line in the data file
DO WHILE NOT EOF(DATAFILENUM)
  INPUT @DATAFILENUM, bubbler, count, temperature, delay ' Get the data
  LINE INPUT @DATAFILENUM, rest$ ' Read the rest of the line

' Check the temp to see if its close enough to the specified temperature.
IF ABS(temperature—selecttemp) < THRESHOLD AND bubbler = SELECTBUB THEN
  IF bubbler = 1 THEN
  PRINT 'Got bubbler 1.'

  ' We open a new output file for this temperature only if one
  has't been opened before. 'resetbubbl.flag' can only be
  TRUE if this data line is the first one of the correct
  temperature.
  IF resetbubbl.flag = TRUE THEN
      CLOSE(BUBB1RAWNUM)
      outfile$ = MID$(str$(selecttemp),2) + "B1.dat"
      OPEN outfile$ FOR OUTPUT AS BUBB1RAWNUM
  END IF
  PRINT BUBB1RAWNUM, USING D '; bubbler;
  PRINT BUBB1RAWNUM, USING . *; temperature;
  PRINT BUBB1RAWNUM, USING 'MAD '; delay;
  PRINT BUBB1RAWNUM, rest$ 

  ' Don't reset the accumulated statistics after the first pass
  resetbubbl.flag = FALSE

  ELSEIF bubbler = 2 THEN
  PRINT "Got bubbler 2."

  ' We open a new output file for this temperature only if one
  hasn't been opened before. 'resetbubb2.flag' can only be
  TRUE if this data line is the first one of the correct
  temperature.
  IF resetbubb2.flag = TRUE THEN
      CLOSE(BUBB2RAWNUM)
      outfile$ = MID$(str$(selecttemp),2) + "B2.dat"
      OPEN outfile$ FOR OUTPUT AS BUBB2RAWNUM
  END IF
  PRINT BUBB2RAWNUM, USING D '; bubbler;
  PRINT BUBB2RAWNUM, USING . *; temperature;
  PRINT BUBB2RAWNUM, USING 'MAD '; delay;
  PRINT BUBB2RAWNUM, rest$

  ' Don't reset the accumulated statistics after the first pass
  resetbubb2.flag = FALSE
  END IF
END IF
LOOP
END IF
LOOP
NEXT T
CHDIR "D:\GB"  
END
Program segment allows recovery from some errors.

HANDLER:

errnum = ERR

IF errnum = 62 THEN  ' Input past end error
  PRINT "Input past end. Resuming with next file."
  RESUME READFILE
ELSE
  ERROR errnum
  ON ERROR GOTO 0
END IF