

# ornl

**OAK RIDGE  
NATIONAL  
LABORATORY**

**MARTIN MARIETTA**

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

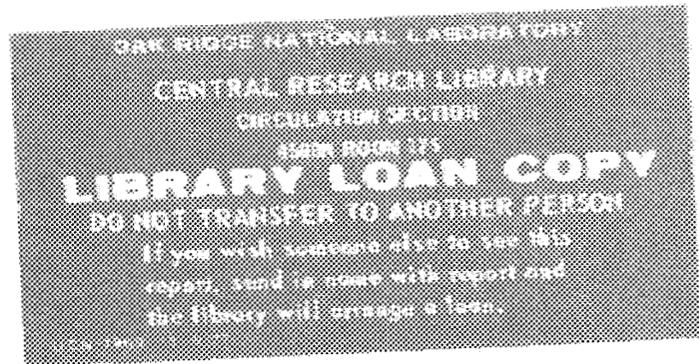


3 4456 0275029 9

ORNL/TM-10657

## A Microcomputer-Based Averaging Flowmeter Using FORTH Programming Language

W. B. Jatko



OPERATED BY  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
FOR THE UNITED STATES  
DEPARTMENT OF ENERGY

Printed in the United States of America. Available from  
National Technical Information Service  
U.S. Department of Commerce  
5285 Port Royal Road, Springfield, Virginia 22161  
NTIS price codes—Printed Copy: A05 Microfiche: A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-10657  
Dist. Category UC-506

Instrumentation and Controls Division

A MICROCOMPUTER-BASED AVERAGING FLOWMETER  
USING FORTH PROGRAMMING LANGUAGE\*

W. B. Jatko

Date Published: April 1988

\*Thesis presented for the Master of Science Degree,  
University of Tennessee, Knoxville, 1987.

Prepared by the  
OAK RIDGE NATIONAL LABORATORY  
Oak Ridge, Tennessee 37831  
operated by  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
for the  
U.S. DEPARTMENT OF ENERGY  
under Contract No. DE-AC05-84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES



3 4456 0275029 9



## CONTENTS

	Page
1. INTRODUCTION . . . . .	1
1.1 Background Information . . . . .	1
1.2 System Specifications . . . . .	1
1.3 Alternative Methods . . . . .	2
1.4 Summary . . . . .	2
2. DEVELOPMENT OF THE MODEL . . . . .	3
2.1 Bubbler System . . . . .	3
2.2 Vessel . . . . .	4
2.3 Differential Pressure Cells . . . . .	5
2.4 Current-to-Voltage Conversion . . . . .	5
2.5 Analog-to-Digital Conversion . . . . .	6
2.6 Computer Flow Equation . . . . .	6
2.7 Error Estimates . . . . .	8
3. HARDWARE DEVELOPMENT . . . . .	10
3.1 System Requirements . . . . .	10
3.2 Power Supply Frame . . . . .	10
3.3 Single-Board Computer . . . . .	12
3.4 Analog I/O Board . . . . .	12
3.5 Alarm Interface . . . . .	15
3.6 Display Interface . . . . .	15
3.7 Reset/Analog Interface Board . . . . .	15
4. SOFTWARE DEVELOPMENT . . . . .	20
4.1 Preliminary Considerations . . . . .	21
4.2 Development of Least-Squares Fit . . . . .	22
4.3 Program Development . . . . .	26
4.4 System Initialization . . . . .	28
4.5 Data Array . . . . .	28
4.6 RESET Service Routine . . . . .	28
4.7 TIMER Service Routine . . . . .	30
5. EXPERIMENTAL RESULTS . . . . .	34
5.1 Analog Input Filter . . . . .	34
5.2 Verification of Model . . . . .	34
5.3 Scale Factor . . . . .	34
5.4 Least-Squares Algorithm . . . . .	35
5.5 Statistical Sampling . . . . .	35
5.6 Operational Tests . . . . .	37
5.7 Transient Response . . . . .	38
5.8 Long-Term Stability . . . . .	40

6. SUMMARY AND CONCLUSIONS . . . . .	41
REFERENCES . . . . .	42
APPENDIX A . . . . .	43
APPENDIX B . . . . .	50

## LIST OF FIGURES

Figure	Page
1. Diagram of the containment vessel . . . . .	4
2. Flowmeter block diagram . . . . .	11
3. Single-board computer block diagram . . . . .	13
4. Analog-to-digital converter block diagram . . . . .	14
5. Alarm interface block diagram . . . . .	16
6. Display interface block diagram . . . . .	16
7. RESET interface diagram . . . . .	18
8. Analog input filter for typical channel . . . . .	19
9. Arbitrary collection of data points illustrating a least-squares fit . . . . .	23
10. Arbitrary data points after translating origin . . . . .	25
11. Data points for function $y = 2x + 5$ . . . . .	27
12. Flowmeter software block diagram . . . . .	27
13. System initialization flow diagram . . . . .	29
14. Memory map of Channel One data array . . . . .	29
15. RESET interrupt service flow diagram . . . . .	30
16. TIMER interrupt service flow diagram . . . . .	31
17. Effect of analog filter on input signal . . . . .	35
18. Diagram of experimental apparatus . . . . .	36
19. Frequency distribution histogram for data-sampling process . . . . .	38
20. Deviations of calculated flow from measured flow . . . . .	39



## ABSTRACT

A flowmeter was designed and constructed to measure the average fluid flow during a 10-min interval. The instrument used a bubbler system and differential pressure sensors to measure the density-compensated level in a sealed vessel. A microcomputer was used to record a fluid-level histogram and to calculate the slope of the histogram through a linear least-squares regression. The average flow is proportional to the slope of the fluid-level histogram. The average flow rate in four independent vessels was displayed on the front panel.



## 1. INTRODUCTION

### 1.1 BACKGROUND INFORMATION

Instrumentation of a chemical process that uses toxic material poses a difficult problem for the instrument engineer, as he is charged with measuring various phases of the process reliably and accurately. When the chemical process involves a radioactive substance, the process must be adequately shielded and isolated from human operators, requiring complete containment of the process within several inches of lead protection. This method provides protection for the operators, while placing additional constraints on the instrumentation. Any measurement sensor located inside the containment vessel must be reliable because access into the vessel is limited. The sensors must also be immune to radioactive bombardment that might deteriorate performance. The author was asked to design an instrument that would measure average fluid flow within a sealed, radioactive vessel.

At that time, a chemical process was being developed at the Oak Ridge National Laboratory (ORNL) to properly dispose of nuclear power reactor waste products. This particular waste product was a radioactive fluid from a commercial nuclear power facility. In the decontamination process, the fluid was solidified in metal cylinders. When filled, the end caps of the cylinders were welded in place and the cylinders were stored in a designated "safe" waste disposal area. For solidification to occur, the chemical reaction rate must be precisely controlled, and this was done by controlling the mass flow of reactants and the solution from a large reservoir into the solidification vessel. Since the reservoir was to be equipped with a bubbler system to measure the fluid level, a method of measuring the flow rate was devised using these fluid-level sensors. The bubbler system has the desired immunity to radioactive attack, is accurate and reliable, and can be used to measure the density of the fluid. The fluid density is uniform throughout the reservoir, but its measurement is variable with a specific gravity between 1 and 2. The density is important and must be taken into account in controlling the mass flow.

The method chosen to measure the density of the fluid uses a single-board computer and a data acquisition board to measure and chart trends in the reservoir level. By calculating the level changes over a period of time, an estimate of average mass flow over that time period can be made.

### 1.2 SYSTEM SPECIFICATIONS

The flow measurement should have an accuracy of  $\pm 5\%$ . The nominal flow is 9.0 mL/min  $\pm 5$  mL/min from a 7500-mL reservoir, although it is desired to measure 9.0 mL/min  $\pm 0.4$  mL/min. There are four independent channels, with each channel having a separate reset, digital flow indicator, and

over/under flow alarms. The flow rate display should have a 0- to 29.9-mL/min range with a 0.1-mL/min resolution.

### 1.3 ALTERNATIVE METHODS

Although many methods of measuring instantaneous flow rate are more precise, they were deemed unsatisfactory because they require that electrical sensors be placed in the restricted access area. The bubbler system permits all electrical components to be mounted outside the restricted area for maintenance access. Because maintenance must be minimized, the bubbler system was chosen as the best alternative.

### 1.4 SUMMARY

The method chosen uses a bubbler system with differential pressure transducers to measure the fluid level and density within a sealed containment reservoir. A single-board computer using a resident FORTH programming language was selected to record fluid-level histograms and to calculate the slope of the histogram. The computer is a P-FORTH model made by Peopleware Systems, Inc., and is an STD bus design using a 6801 microprocessor and a 6522 peripheral interface adapter. A 16-channel analog-to-digital converter, Model RTI-1260 from Analog Devices, was selected for the data acquisition board. By time multiplexing, the single P-FORTH computer monitors four channels of flow, and level and density data are taken at 6-s intervals and stored in a data array. The computer performs a linear least-squares regression on the level/density points to estimate an average flow rate.

The chemical reaction rate was slow enough to allow the flow rate to be averaged over a 10-min period. The long measurement interval is required to allow the statistical accumulation of data, with precision improved by the taking of large amounts of data. The software uses 100 data points per measurement, which increases precision by an order of magnitude over a single-point measurement.

## 2. DEVELOPMENT OF THE MODEL

The radioactive fluid instrumented was waste material from a commercial nuclear power reactor that was being processed at ORNL. The solution was solidified within metal cylinders that could be sealed and stored. The solidification process required that the solution be pumped at a controlled rate from a containment reservoir vessel into the solidification vessel where the chemical reaction occurred. Figure 1 is a schematic diagram of the containment vessel.

### 2.1 BUBBLER SYSTEM

Two metal tubes deliver a small flow of air from the main air supply into the vessel. One tube extends to the bottom of the vessel and the other to within distance  $D$  of the bottom. The pressure in the bottom of the tube is just enough to force an air bubble from the tube; therefore, the pressure in the tube is nearly the same as the hydrostatic pressure of the fluid at the tubing outlet. The air velocity is small, so pressure drops along the tubing are neglected. A differential pressure-to-current transducer generates a 4- to 20-mA current proportional to the pressure difference across the pressure sensor. One pressure cell, connected from the level bubbler tube to the top of the vessel, measures the pressure of the fluid in the vessel and is proportional to the fluid height,  $h$ . The pressure equation for a static fluid is well known:

$$P = \rho gh \quad , \quad (2.1)$$

where  $P$  is the pressure,  $\rho$  is the fluid density, and  $g$  is the gravitational constant, which leads to the equation for the level tube:

$$P_L = \rho gh \quad . \quad (2.2)$$

The second pressure cell is connected between the level tube and the density tube and measures the pressure drop across distance  $D$ . Equation (2.1),  $P = \rho gh$ , still applies, but  $h$  is known to be equal to  $D$ , and we can write

$$P_D = \rho gD \quad . \quad (2.3)$$

Since  $\rho$  and  $g$  are the same for both Eqs. (2.2) and (2.3), we can divide and cancel the  $\rho g$  terms. This yields  $P_L/P_D = h/D$ , which can be written as

$$h = \frac{P_L D}{P_D} \quad . \quad (2.4)$$

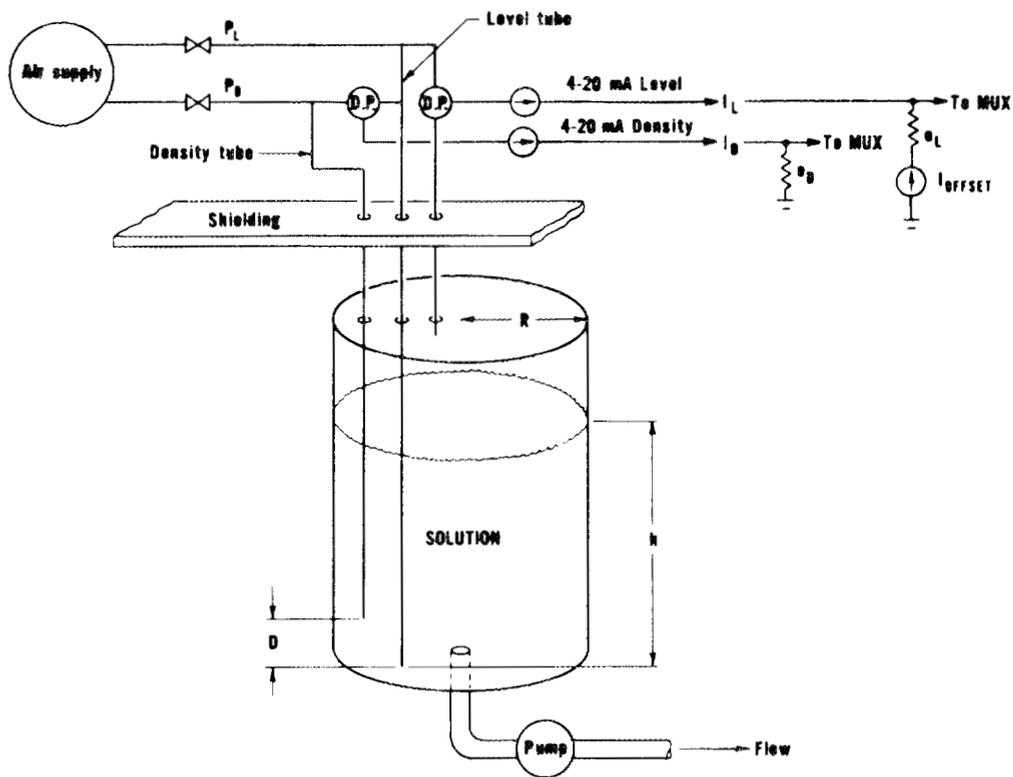


Fig. 1. Diagram of the containment vessel.

## 2.2 VESSEL

The fluid flow is the time rate of change in volume and can be written mathematically as

$$F = \frac{\partial}{\partial t} (\text{volume})_{\text{fluid}} \quad (2.5)$$

The volume of fluid in a cylinder is

$$V = \pi R^2 h \quad (2.6)$$

Taking the derivative of Eq. (2.6) and assuming only time variance,

$$\frac{\partial}{\partial t} V = \pi R^2 \frac{d}{dt} (h) \quad (2.7)$$

Substituting Eqs. (2.7) and (2.4) into Eq. (2.5),

$$F = \pi R^2 D \left( \frac{d}{dt} \right) \left( \frac{P_L}{P_D} \right) \quad (2.8)$$

### 2.3 DIFFERENTIAL PRESSURE CELLS

The differential pressure cells convert pressure to current. The cell is designed to give a linear relationship between pressure and current in the form

$$I = mP + b \quad , \quad (2.9)$$

where  $I$  is the output current,  $m$  is a scale factor, and  $b$  is an offset. Both  $m$  and  $b$  can be chosen to suit the application.

In this application, the pressure cell connected to the level probe is calibrated to provide 4 mA for zero pressure and 20 mA for the maximum head pressure,  $H_L$ . Since the solution has a maximum density twice that of water,  $H_L$  must be twice the vessel volume. Thus, the 7.5-L vessel might have a potential fluid pressure equivalent to 15.0 L of water. Using the proportionality  $0 - H_L : 4 - 20$ , we can derive an equation for the pressure-to-current transfer function in the level probe

$$I_L = \left( \frac{16}{H_L} \right) P_L + 4 \quad , \quad (2.10)$$

where  $I_L$  is in milliamperes.

The density probe transfer function is different from the level probe. The density probe has a minimum pressure equal to the height of a column of water  $D$  in. high. With a fluid of specific gravity 2, the maximum pressure is equivalent to a height of  $2D$ . Thus, the density pressure cell has an operating range of  $D$  to  $2D$ . This leads to the proportionality  $D - 2D : 4 - 20$ , which can be written as

$$I_D = \frac{16}{D} P_D - 12 \quad . \quad (2.11)$$

### 2.4 CURRENT-TO-VOLTAGE CONVERSION

The analog-to-digital converter (ADC) has a 5-V full-scale range. The range of the ADC integrated circuit is 10 V full scale, but a variable-gain preamplifier is available to allow the use of the 5-V full-scale input.

Normally the current-to-voltage conversion is done by using a 250- $\Omega$  resistor in series with the current, developing a signal voltage from 1 to 5 V. However, this process does not utilize the full 12-bit dynamic range of the ADC, and, consequently, the accuracy of the measurement suffers. In many cases this reduced accuracy is acceptable because of the ease of implementation. In this case, it was decided to use the full 12 bits of accuracy available in the ADC board for the level measurement, as it is the most critical, and to use the simpler, less accurate method for the density measurement. Therefore, an analog circuit was designed to provide a 4-mA offset current and also to provide low-pass filtering. This circuit allowed the 4- to 20-mA

current to develop a 0- to 5-V signal at the ADC. The appropriate transfer equation is

$$e_L = \frac{5}{16} I_L - \frac{5}{4} , \quad (2.12)$$

where  $e_L$  is in volts,  $I_L$  in milliamperes.

The density probe measurement was less critical, and the conventional 250- $\Omega$  resistor method previously described was used. An equation for the density voltage was derived,  $e_D = I_D R_D$ , and, since  $R_D = 250 \Omega$ ,

$$e_D = \frac{I_D}{4} , \quad (2.13)$$

where  $e_D$  is in volts,  $I_D$  in milliamperes.

## 2.5 ANALOG-TO-DIGITAL CONVERSION

The final transfer equation is the analog-to-digital conversion. The ADC has 12 bits of accuracy, allowing  $2^{12}$  or 4096 possible states. Since there is a 5-V full-scale range at the input, we get

$$N^* = \frac{4096}{5} e , \quad (2.14)$$

where  $N^*$  is the digital number outcome of the conversion process. The asterisk is to remind us that  $N$  is a quantized integer and not a continuous number.

## 2.6 COMPUTER FLOW EQUATION

Rearranging Eq. (2.14) and substituting into Eqs. (2.12) and (2.13) gives

$$\frac{5}{4096} N_L^* = \frac{5}{16} I_L - \frac{5}{4} , \quad (2.15)$$

$$\frac{5}{4096} N_D^* = (0.25) I_D . \quad (2.16)$$

Substituting Eqs. (2.10) and (2.11) into Eqs. (2.15) and (2.16), respectively, gives

$$\frac{5}{4096} N_L^* = \left( \frac{5}{16} \right) \left( \frac{16}{H_L} P_L + 4 \right) - \frac{5}{4} , \quad (2.17)$$

$$\frac{5}{4096} N_D^* = \left( \frac{1}{4} \right) \frac{16}{D} P_D - 12 ;$$

simplifying and rearranging,

$$P_L = \frac{H_L N_L^*}{4096} \quad (2.18)$$

$$P_D = \left(\frac{D}{4}\right) \left(\frac{5 N_D^*}{4096} + 3\right) \quad (2.19)$$

Recall the flow equation [Eq. (2.8)],

$$F = \pi R^2 D \frac{d}{dt} \frac{P_L}{P_D} \quad (2.20)$$

The density pressure term is invariant within the measurement period, so it can be moved outside the derivative operation

$$F = \frac{\pi R^2 D}{P_D} \frac{d}{dt} (P_L) \quad (2.21)$$

substituting Eqs. (2.19) and (2.20) into Eq. (2.21),

$$F = \frac{\pi R^2 D}{\left(\frac{D}{4}\right) \left(\frac{5 N_D^*}{4096} + 3\right)} \frac{d}{dt} \frac{H_L N_L^*}{4096} \quad (2.22)$$

The constants are  $H_L$  and 4096 and likewise can be removed from the derivative term:

$$F = \frac{\pi R^2 D H_L}{\left(\frac{D}{4}\right) \left(5 N_D^* + 3 (4096)\right)} \frac{d}{dt} (N_L^*) \quad (2.23)$$

simplifying,

$$F = \frac{4 \pi R^2 H_L}{5 N_D^* + 12288} \frac{d}{dt} (N_L^*) \quad (2.24)$$

Since the intended flow measurement is to be an average flow, we change the derivatives to delta and approximate

$$F \cong \frac{4 \pi R^2 H_L}{5 N_D^* + 12288} \frac{\Delta}{\Delta t} (N_L^*) \quad (2.25)$$

Equation (2.25) is the fundamental equation implemented in the FORTH software algorithm.

One modification of Eq. (2.25) is to place the denominator of the first term within the delta operator and to multiply and divide by 5. This operation was done to simplify the data acquisition software and results in

$$F \cong \frac{4 \pi R^2 H_L}{5} \frac{\Delta}{\Delta t} \frac{5 N_L^*}{5 N_D^* + 12288} \quad (2.26)$$

Since the first term is made of constants, they may be lumped together as a single constant K. We also replace the

$$\frac{5 N_L^*}{5 N_D^* + 12288} \quad (2.27)$$

term by variable Q and write

$$F = K \frac{\Delta Q}{\Delta t} \quad (2.28)$$

Constant K is stored in the computer memory, and  $\Delta Q/\Delta t$  is computed through the linear least-squares regression. The development of the least-squares algorithm is given in Sect. 4.0.

## 2.7 ERROR ESTIMATES

The accuracy of the differential pressure cell is quoted by the manufacturer to be  $\pm 0.25\%$  of full scale.<sup>1</sup> If the nominal vessel volume is considered to be 7500 mL, the bubbler system will measure the volume as 7500 mL  $\pm$  18.75 mL. If we assume an anticipated flow rate of 9.0 mL/min, after 10 min there will be 7410 mL in the vessel. The bubbler system measures this as 7410 mL  $\pm$  18.75 mL. Considering the worst-case condition of the initial measurement at the upper limit of the error band and the final measure at the lower limit of the error band, we obtain measurements of 7518.75 and 7391.25 mL. The change of volume in the vessel appears to be  $\Delta V = 7518.75 - 7391.25 = 127.5$  mL, when actually it is 90 mL, which would be a maximum error of  $\epsilon = [(90 - 127.5)/90] \times 100$  and  $\epsilon = 41.7\%$ .

With an actual flow of 9.0 mL/min, this percentage error is  $\pm 3.75$  mL/min. The 12-bit resolution of the ADC gives a quantizing uncertainty of  $\pm \frac{1}{2}$  least significant bit (LSB) where  $\text{LSB} = 7500/212 = 1.83$  mL/min, then  $\frac{1}{2}$  LSB = 0.62 mL/min.

Assuming the two errors to be uncorrelated, the total error is the square root of the sum of individual errors squared.

$$\epsilon = \sqrt{(3.75)^2 + (0.62)^2} = 3.8 \text{ mL/min} = \pm 42\% \quad (2.29)$$

This calculation shows that the total system bubbler system error dominates the error.

We consider the 3.8-mL/min error to be within two statistical standard deviations, which would include 95% of all occurrences. Therefore, one standard deviation would be  $s = 1.9$  mL/min.

The standard deviation of a sampled group is reduced by the square root of the number of samples,<sup>2</sup>

$$s^* = \frac{s}{\sqrt{n}} \quad (2.30)$$

where  $s^*$  is the data group deviation and  $n$  is the number of data points in the sample group.

To reduce the 1.9-mL/min deviation to less than the desired 0.4-mL/min would require  $n = (1.9/0.4)^2$  samples = 22.6 samples per measurement.

It was decided to use 100 samples in the measurement, with an estimation error of  $\epsilon = 1.9/\sqrt{100} = 0.19$  mL/min, which is within our desired error of  $\pm 0.4$  mL/min.

A physical system model has been developed and an estimation of the anticipated error has been made. The number of data points to use in the least-squares fit was chosen to be 100 in order to reduce the errors of measuring sensors. Although additional errors from component tolerances, aging, and assumptions are also present, they are likely to be negligible, as seen in Eq. (2.28).

### 3. HARDWARE DEVELOPMENT

#### 3.1 SYSTEM REQUIREMENTS

To achieve the desired improvement in accuracy discussed in Sect. 2, a method of taking and holding more than 100 data samples was required. A single-board computer implementation was chosen as the most practical way to acquire data from the pressure sensors and to calculate a slope from the sensor data history. Details of the slope calculations are considered in the section on software development (Sect. 4).

An alternative hardware scheme, using an analog operational amplifier with a hardware divider chip to perform the PL/PD division, was suggested initially. This method uses a sample-hold circuit to store sample values until the next sample is taken. As the time between samples is relatively long (10 min), the sample-hold capacitor can contribute to sampling errors due to voltage discharge during the holding period. This architecture was rejected because of the high potential for sampling errors.

The flow to be measured is slow enough to permit sampling of the vessel level at 6-s intervals. This time interval permitted the use of one data acquisition system for all channels by time multiplexing.

Requirements for the system included four separate channels of flow measurement with a three-digit display, 120-V ac high/low flow alarms, and a reset for each channel.

A flow measurement system was assembled using a single-board computer, analog-to-digital conversion board, display interface board, alarm interface board, reset/analog interface board, and power supply frame. The system block diagram is shown in Fig. 2.

#### 3.2 POWER SUPPLY FRAME

The power supply frame and card cage for the STD bus components is a Pro-Log Model 701B. This model was chosen because it has enough card slots to allow future expansion and comes in a 19-in. rack-mountable frame. This mounting was desirable as the instrument would be installed in a process control room. The 19-in.-wide cabinet also provided enough front-panel space to mount digital display meters and reset buttons for all four channels. With this cabinet, the four-channel flow monitoring system can be contained in one integral package. The Pro-Log cabinet also provided sufficient internal space to mount an auxiliary power supply for the pressure-to-current transmitters. The STD card cage holds 13 cards connected on the STD bus plane.

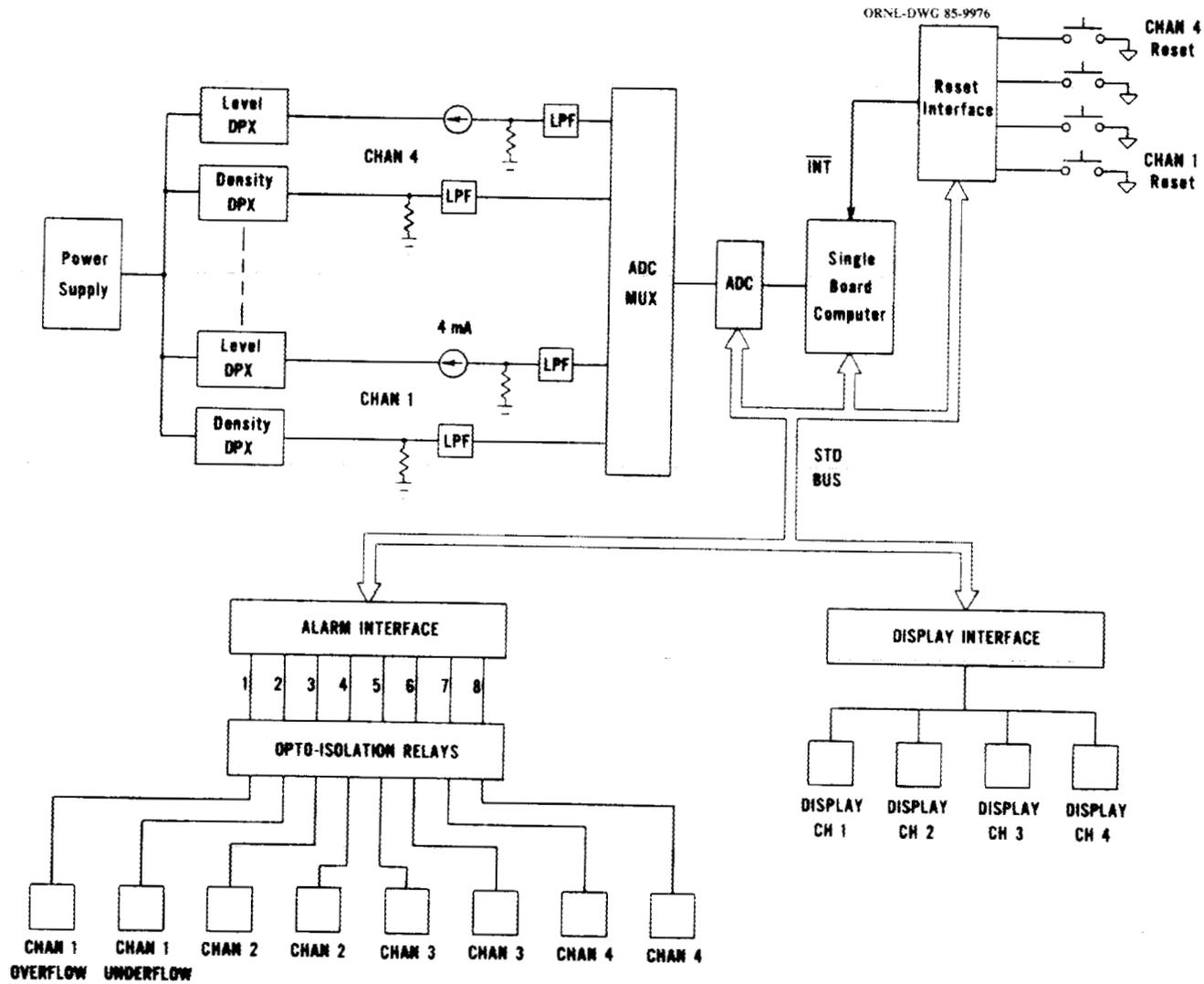


Fig. 2. Flowmeter block diagram.

### 3.3 SINGLE-BOARD COMPUTER

The heart of the flow monitor is the P-FORTH card by Peopleware Systems, Inc. This card uses a Motorola MC 6801 microprocessor and a 6522 peripheral interface adapter (PIA). The P-FORTH card has 2 Kbytes of random-access memory (RAM) and 6 Kbytes of electrically erasable read-only memory (EEROM). The FORTH programming language is resident on the card in 8 Kbytes of read-only memory (ROM). There are also 16 input/output (I/O) lines available through the PIA and a serial RS-232 port for communications via a CRT terminal. Interrupts may be generated by the PIA in response to various external events. A block diagram of the single-board computer card is shown in Fig. 3.<sup>3</sup>

The MC6801 microprocessor uses a memory-mapped I/O scheme. However, the STD bus supports an I/O mapped region for use with Z-80 type microprocessors. The P-FORTH card handles this situation by decoding a block of memory from 0100-01FF hexadecimal as the I/O region. Memory accesses in this region are decoded by the P-FORTH circuitry as an I/O request, allowing 256 possible I/O locations in the system.

A convenient feature of the P-FORTH card is the use of the EEROM memory. When a FORTH application program is compiled, it is placed in the EEROM space. Once application software has been written and tested satisfactorily, the P-FORTH card can be switched from a development mode to a target application mode. The application software can be vectored to begin execution upon power-up, and it allows the designer to use the same card both as a development station and a target machine.

The flow monitor uses the two timers on the 6522 PIA to generate the sample command. The vessel volume is sampled every 6 s. Since four channels are time multiplexed, a sample command is issued at 1.5-s intervals. A software algorithm maintains proper channel sequencing.

Another interrupt is generated on the reset/analog interface card. The reset interrupt indicates that one or more channels are requesting reset. The MC6801 polls the 6522 adapter interrupt register to determine if an interrupt request is from a reset command or a sampling command. The sampling command has the higher priority in the polling. More detailed consideration of the software is given in Sect. 4.

### 3.4 ANALOG I/O BOARD

The vessel volume and density are sampled and quantified on the analog I/O board, which is an Analog Device Model RTI-1260. The RTI-1260<sup>4</sup> (Fig. 4) provides 16 single-ended or 8 differential inputs that are converted to 12 bits. A programmable gain amplifier accepts input signals up to 10 V. The sample-hold circuit and AD574 analog-to-digital converter provide throughput rates of up to 25,000 channels/s. The analog-to-digital conversion time is 25  $\mu$ s. The

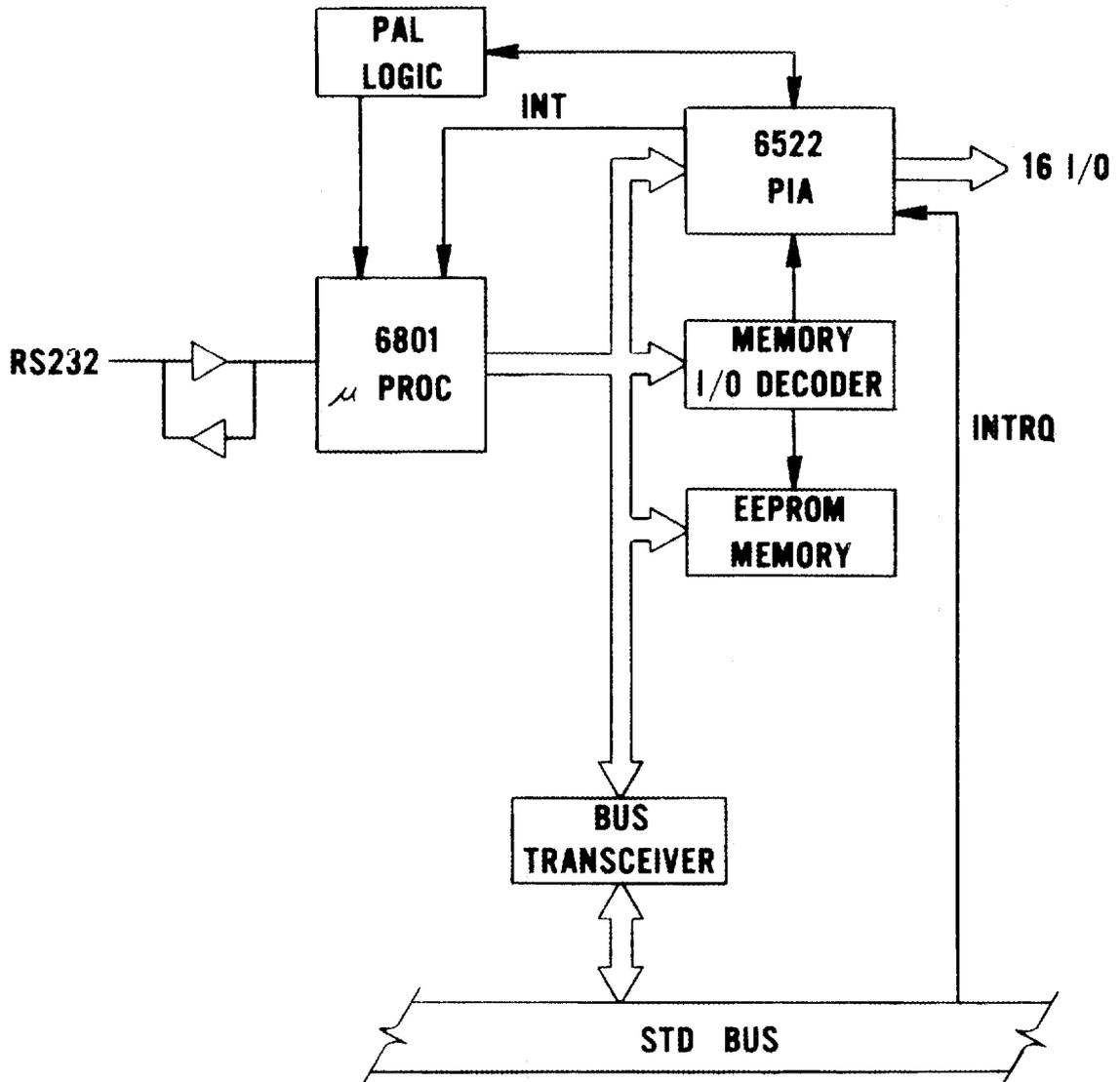


Fig. 3. Single-board computer block diagram.

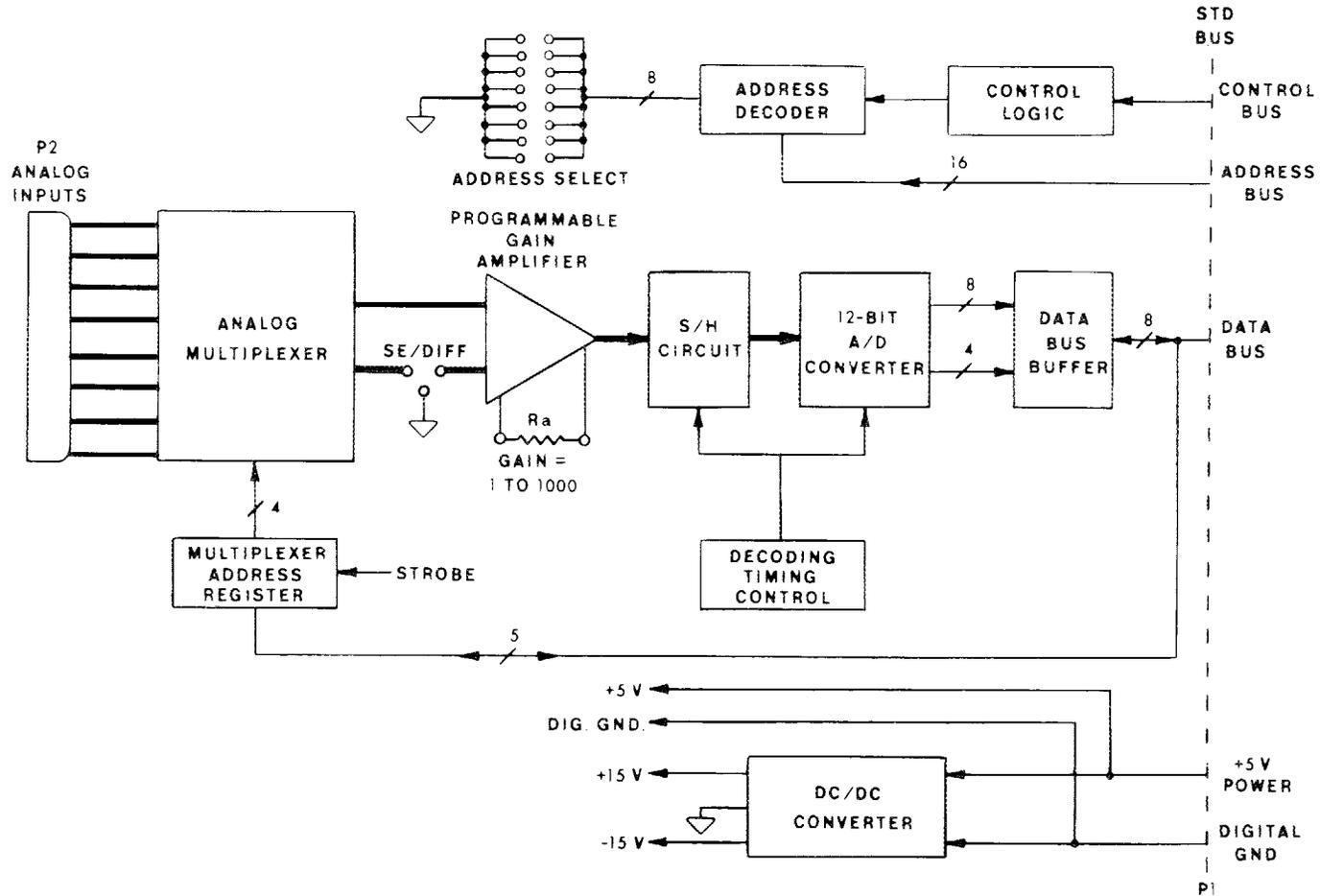


Fig. 4. Analog-to-digital converter block diagram.

RTI-1260 has a dc-to-dc converter on board that allows it to be powered from the 5-V dc on the STD backplane.

### 3.5 ALARM INTERFACE

The flow monitor is required to provide contact closures to indicate flow levels above or below prescribed limits. The P-FORTH card compares the current flow rate with the upper/lower limits stored in memory.

A Pro-Log Model 7507 interface card is used to latch the status of each channel into the I/O region of memory. The Model 7507 card is interfaced to an OPTO-22 optical isolation relay card,<sup>5</sup> which holds eight optically isolated solid-state relays. Each channel has one relay to indicate overflow and underflow. The relay contacts activate supervisory warning lamps in the main control room. The eight alarms are mapped directly to the 8-bit data word written to the Model 7507. The selected alarm is activated by setting the corresponding bit in the data word high and writing to the Model 7507 input port. The Model 7507 has internal latches to hold the status until the alarm is cleared.<sup>6</sup> Figure 5 shows the Model 7507 diagram.

### 3.6 DISPLAY INTERFACE

Each channel in the flow monitor has a four-digit display to indicate average flow in milliliters per minute. The displays use binary coded decimal (BCD) inputs for each digit, so 16 input lines are required per channel. A Pro-Log Model 7602 output port card (Fig. 6) provides the interface between the P-FORTH card and the four display modules. The Model 7602 has 64 output lines and maps into the I/O region of the P-FORTH card.<sup>7</sup> The Model 7602 has eight output ports to allow interfacing to the outside world. Each display module is connected to two output ports on the Model 7602, and this connection allows each display module to have an address in the I/O region.

Flow data from the P-FORTH card are converted to BCD format and then written to the proper channel display module. The Model 7602 latches the BCD values into the channel output ports, where the display value is held until the next update cycle.

### 3.7 RESET/ANALOG INTERFACE BOARD

The reset/analog interface card provides the auxiliary functions of analog, filtering the signal currents from the pressure sensors and generating an interrupt request for a channel reset. The reset-analog card used is a Pro-Log Model 7904,<sup>8</sup> a blank wire-wrap card with an I/O

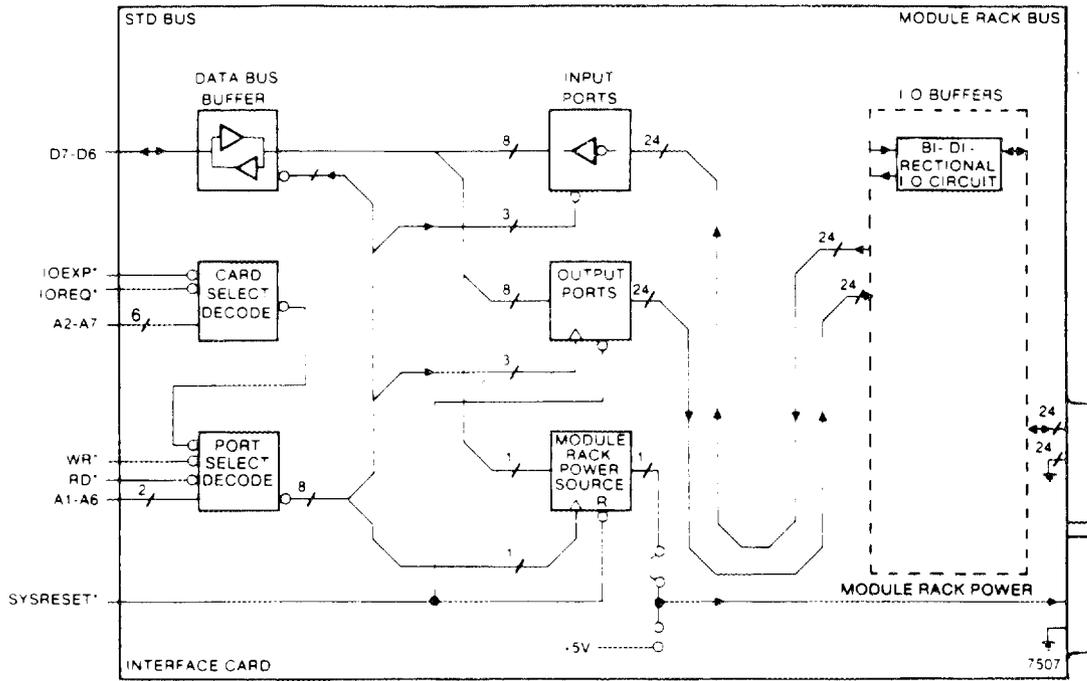


Fig. 5. Alarm interface block diagram.

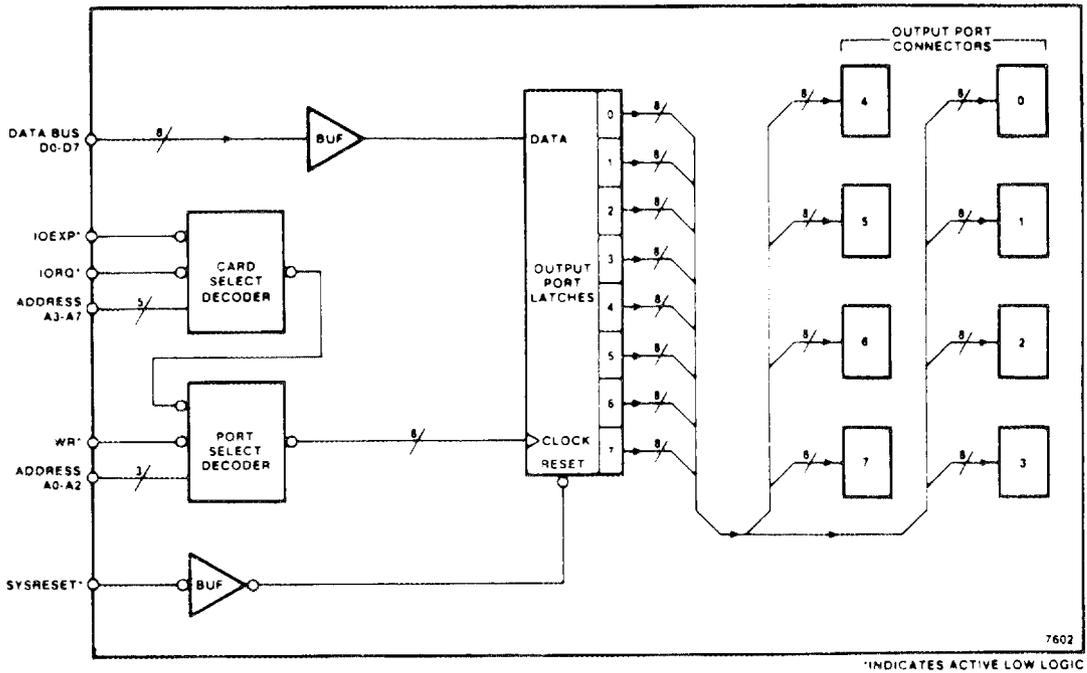


Fig. 6. Display interface block diagram.

decoder provided, allowing an external I/O function to use the onboard decoder and bus transceivers in accessing the STD bus.

The reset section of the card is shown in Fig. 7. The front panel pushbuttons are used to trigger a 7474 latch after being "debounced" by the MC14490. The latches are connected to an 8-to-3 encoder, which provides a binary output of the channel number to be reset and generates an interrupt request on the STD bus pin 44. The interrupt request line (INTRQ\*) will set an interrupt flag in the Model 6522 PIA chip on the P-FORTH card. The Model 6801 microprocessor reads the reset I/O port to determine the proper channel to be cleared, and the interrupt condition on the Model 7904 reset-analog card is cleared by writing a dummy word to the port. The analog section for a typical channel is shown in Fig. 8.

The density signal is developed across a 250- $\Omega$  resistor. The capacitors in parallel form a 125-Hz low-pass filter, which acts to remove noise and other high-frequency artifacts from the input signal.

The level signal is filtered similarly and also has a 4-mA current source to provide an offset current. The offset current is used to translate the 4- to 20-mA signal to 0 to 16 mA. This translation is done to utilize the full dynamic range of the 12-bit analog-to-digital converter (ADC). The ADC input range is 0 to 5 V; the 4- to 20-mA current would develop a 1- to 5-V signal as seen in the density channel, resulting in 1 V of the full range of the ADC not being utilized. This discrepancy is undesirable in the level channel where accuracy is most important. The density range of the fluid being measured is small enough to permit the loss of dynamic range in conversion.

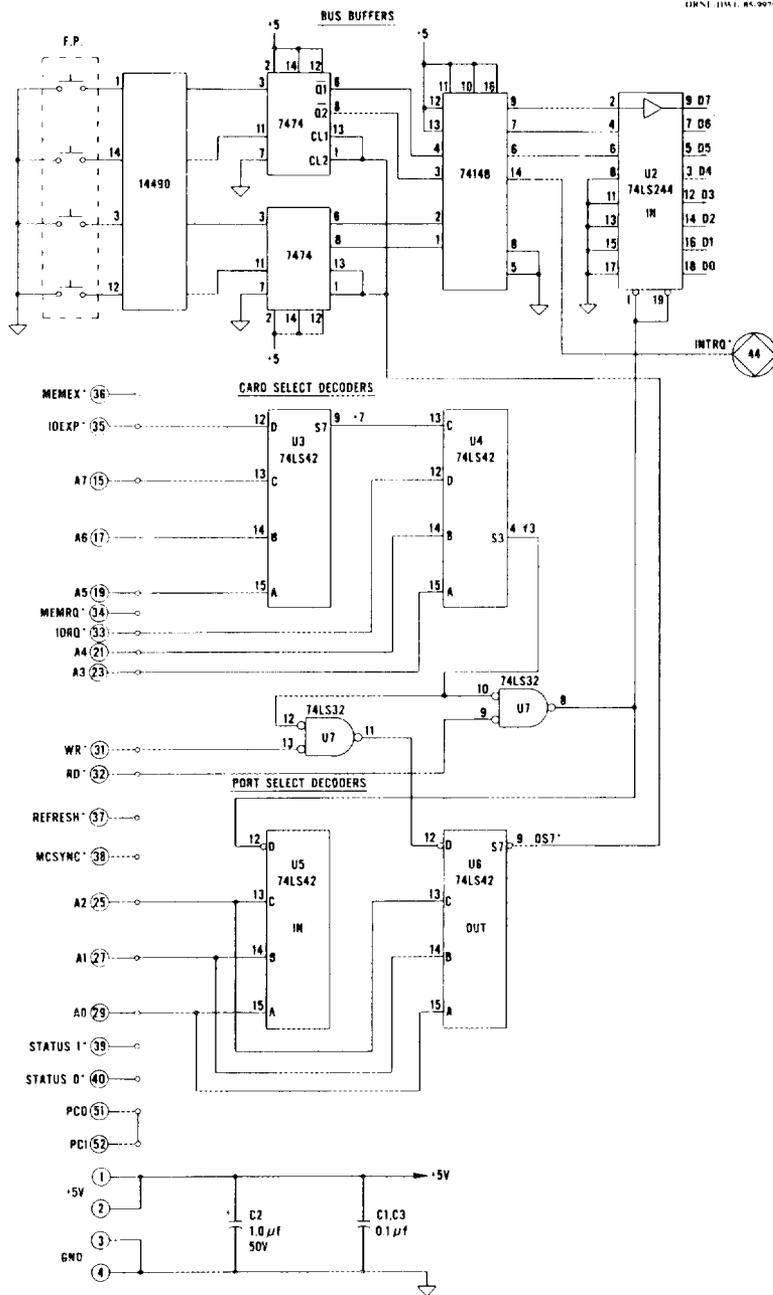


Fig. 7. Reset interface diagram.

ORNL-DWG 85-9980

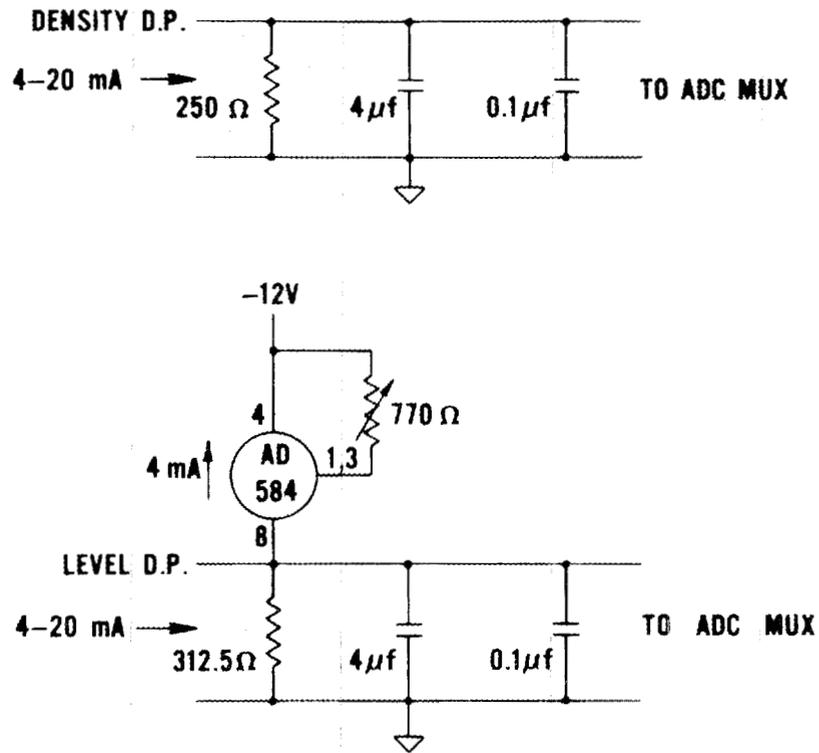


Fig. 8. Analog input filter for typical channel.

#### 4. SOFTWARE DEVELOPMENT

The FORTH programming language was chosen for this application of instrumentation because of its advantages in real-time instrument systems. FORTH is a modularized, structured, extensible programming environment originally designed for real-time process control applications. The primary advantages of FORTH are its interactive nature, flexibility, and efficient use of hardware resources.

The FORTH language is organized as a dictionary in memory. The dictionary is comprised of "words," which are the fundamental unit of the language. A word is any string of characters separated by spaces. Each word can be called from the terminal and executed in an interactive mode, or it can be executed as a part of a compiled algorithm. Because FORTH is a stack oriented language, all parameters and data are passed through the stack between the words of the program.

A new word can be created, defined in terms of previously defined words, and added to the dictionary. Once in the dictionary, the newly defined word is available for use by the programmer. The capability to define new words and add them to the language gives FORTH its extensibility, which allows the programmer to create a language uniquely designed for each application. FORTH has a precompiled kernel of rudimentary words that handle such tasks as stack manipulation, program control, terminal I/O, and other functions. The applications programmer typically begins creating "application words" from predefined words in the FORTH kernel, and as more application words are created, they may be combined to perform more complex functions. This process can be continued until the application program is a single word.

As mentioned previously, FORTH has an interpreter that can be used to immediately test newly created words from the terminal. This ability is of tremendous value to the programmer as it allows each word to be tested and debugged in an interactive mode. As an example, the use of an ADC requires a series of commands from a controller that instruct it to perform a conversion and return the results of that conversion. A FORTH word can be defined that reads a value from the ADC and prints the results on the CRT screen. The word can then be called from the terminal. If the conversion is not performed correctly, the word can be changed and retried until it executes in the desired fashion. The immediacy of the trial-and-error process can reduce program development time as much as a factor of 10 compared with conventional languages.<sup>9</sup>

One feature of FORTH that makes it attractive in small instruments is the minimal amount of memory it requires. The typical FORTH system will include a compiler, assembler, text editor, and utility functions; yet all these occupy less than 6 Kbytes of memory. FORTH applications require less memory than equivalent assembly language code and reduce

speed only 25%. For time-critical areas of the program, the programmer may elect to use the built-in assembler to write at the code level.

#### 4.1 PRELIMINARY CONSIDERATIONS

The average flow rate was developed as described in Sect. 2. Equation (2.28) shows that the average flow rate is equal to the product of a scaling constant and the time derivative of the vessel volume. The scaling constant,  $K$ , is determined by the physical parameters of the vessel and bubbler system and will remain constant for any given vessel. The time derivative of the vessel volume is dynamic and must be known to determine the average flow rate.

The time derivative of vessel volume can be estimated by several methods in the computer. One method is to subtract the final level from the initial level and divide by the time interval, giving a  $\Delta V/\Delta t$  over the interval  $T$ . While this method is easily understood, it lacks precision due to the uncertainty of the volume measurement. As shown in Sect. 2, the accuracy of any volume sample is within  $\pm 18$  mL. A randomness of 36 mL can lead to variations of up to 50% in the calculated flow rate. In order to minimize the effect of data-point uncertainty, an array is allocated in the computer memory to store vessel volume data. As a new data point is taken, it is added to the array. A time interval of 10 min was selected to allow a sufficiently measurable volume change, which is necessary because the nominal flow rates in the solidification process are small. Consequently, the vessel volume change is imperceptible without a long collection interval. As shown previously, the accuracy of the flow rate can be improved by statistical averaging. The discussion in Sect. 2 shows that by using 100 samples of data, the precision can be improved by a factor of 10. The method chosen to implement all the aforementioned concepts was a linear regression of the data array.

In order to get 100 data points spanning the desired 10-min interval, it is necessary to acquire a new sample at 6-s intervals. As each new sample is acquired, it is added to the array and a linear regression is applied to the array. Since each data point is a vessel volume sample and the sampling interval is fixed, the array is a two-dimensional graph of volume vs time.

By using a linear regression on the entire array, the best straight-line fit is found. The straight line is defined to be the average flow rate during the array time interval. The linear regression is implemented by minimizing the error squared or least-squares fit. Development of the least-squares fit algorithm is explained in the following section.

## 4.2 DEVELOPMENT OF LEAST-SQUARES FIT

The slope of the normalized level vs time graph is determined through a least-squares regression, which forces a straight line to fit a collection of points by minimizing the deviations between the generated straight line and the points. Figure 9 shows an arbitrary set of points that represent the sampled fluid level,  $y$ . The horizontal axis is the time in minutes,  $t$ . Let the desired straight line be  $F$ , where  $F$  is of the form

$$F = mt + b \quad . \quad (4.1)$$

The error between the straight line and any  $y$  is

$$E = F - y = mt + b - y \quad . \quad (4.2)$$

The errors may be positive or negative, so we square the error term to constrain  $E$  to positive values:

$$E^2 = (mt + b - y)^2 \quad . \quad (4.3)$$

We wish to find values for  $m$  and  $b$  that will minimize the total errors from  $F$ , so we take the partial derivative of the sum of all  $E^2$  terms and set the result equal to zero.

The sum of all error terms is

$$\sum E^2 = \sum (mt_i + b - y_i)^2 \quad , \quad (4.4)$$

where the subscript  $i$  is to indicate discrete values for  $t$  and  $y$ . Take the partial with respect to  $m$  and  $b$ ,

$$\frac{\partial \sum E^2}{\partial m} = 2 \sum (mt_i + b - y_i)(t_i) \quad , \quad (4.5)$$

$$\frac{\partial \sum E^2}{\partial b} = 2 \sum (mt_i + b - y_i) \quad , \quad (4.6)$$

and set these two terms equal to zero:

$$\sum (mt_i^2 + bt_i - y_it_i) = 0 \quad , \quad (4.7)$$

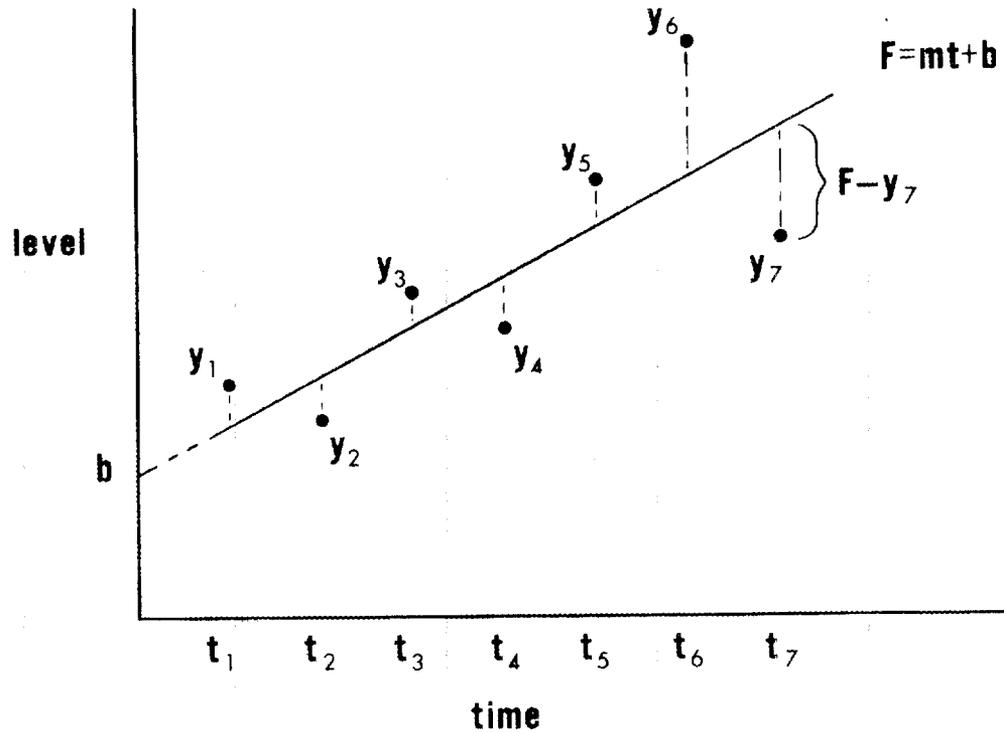


Fig. 9. Arbitrary collection of data points illustrating a least-squares fit.

$$\sum (mt_i + b - y_i) = 0 \quad (4.8)$$

Expand and recognize that  $m$  is a constant:

$$m \sum t_i^2 + \sum bt_i = \sum y_i t_i \quad (4.9)$$

$$m \sum t_i + \sum b = \sum y_i \quad (4.10)$$

If there are  $n$  number of points, the summation of  $b$  is  $nb$ , and we can write

$$m \sum t_i^2 + \sum bt_i = \sum y_i t_i \quad (4.11)$$

$$m \sum t_i + nb = \sum y_i \quad (4.12)$$

We can now separate terms to find expressions for  $m$  and  $b$ :

$$m = \frac{\sum y_i t_i - \sum b t_i}{\sum t_i^2} , \quad (4.13)$$

$$b = \frac{\sum y_i - m \sum t_i}{n} . \quad (4.14)$$

If we impose a few physical constraints, the above expressions can be simplified considerably and implemented by numerical techniques.

The start of time is arbitrary; therefore, the calculation of  $m$  is unaffected by the placement of the origin along the time axis. If we force the array to have only an odd number of points, the origin can be placed in the center of the array. Thus, if there were seven points, we could choose to define the fourth point  $t = 0$ . This would mean that points 1, 2, and 3 would have been taken during negative time in a mathematical sense, but this will not affect the results (Fig. 10).

It can be seen from Figs. 9 and 10 that the data points have not been changed; only the location of the origin changed. The advantage of this mathematical sleight of hand can be seen by re-examining Eq. (4.13).

Since  $t_i$  represents the sampling interval (which is constant),  $t_i$  is a series of integers in both directions; therefore, the sum of  $t_i$  is the sum of a positive series of integers and an equal negative series:  
 $\sum t_i = \sum t_i + \sum t_{-i} = \sum t_i - \sum t_i = 0$ .

Recognizing this, we can eliminate the  $\sum t_i$  terms in Eqs. (4.13) and (4.14), resulting in

$$m = \frac{\sum y_i t_i}{\sum t_i^2} , \quad (4.15)$$

and

$$b = \frac{\sum y_i}{n} . \quad (4.16)$$

The right side of Eq. (4.16) can be seen to be the arithmetic average of the data points,  $b = \bar{y}$ .

The numerical algorithm for  $m$  is developed by recalling that the sampling rate for this application is 0.1 min. This observation lets us rearrange the numerator in Eq. (4.15) to be

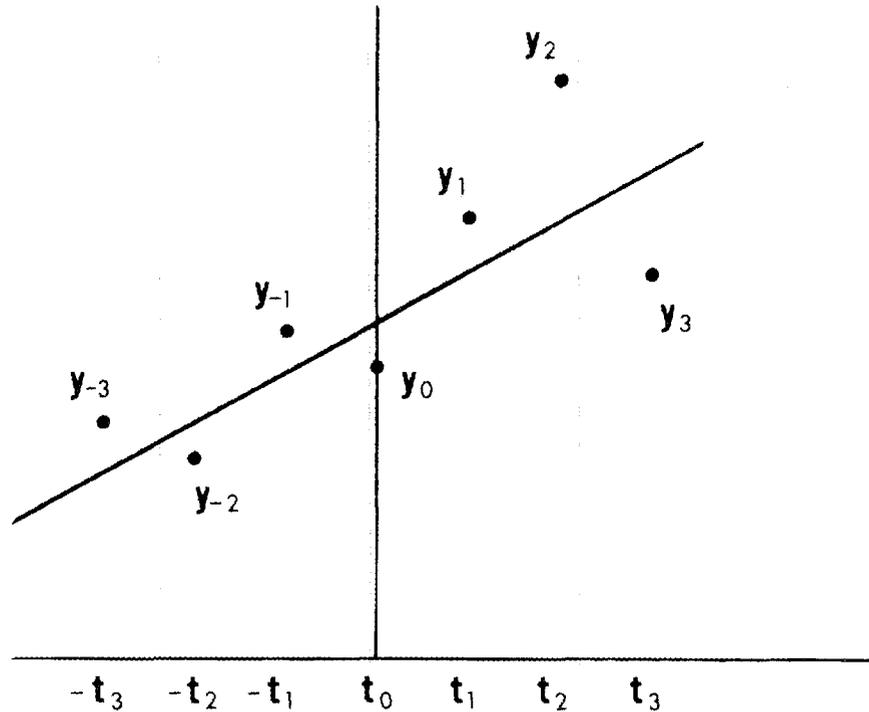


Fig. 10. Arbitrary data points after translating origin.

$$\begin{aligned} \sum y_i t_i &= y_{-3} t_{-3} + y_{-2} t_{-2} + y_{-1} t_{-1} \\ &\quad + y_0 t_0 + y_1 t_1 + y_2 t_2 + y_3 t_3 \end{aligned}$$

Since  $t_0 = 0$ , the middle term on the right side may be eliminated. Using the numerical values for  $t$ ,

$$\begin{aligned} \sum y_i t_i &= (y_{-3})(-0.3) + (y_{-2})(-0.2) + (y_{-1})(-0.1) \\ &\quad + (y_1)(0.1) + (y_2)(0.2) + (y_3)(0.3) \end{aligned}$$

This can be factored to give

$$\sum y_i t_i = (0.3)(y_3 - y_{-3}) + (0.2)(y_2 - y_{-2}) + (0.1)(y_1 - y_{-1})$$

The denominator of Eq. (4.15) is seen to be the sum of negative terms squared and positive terms squared or simply twice the positive terms squared,

$$\sum t^2 = 2(t_1^2 + t_2^2 + t_3^2) \quad . \quad (4.17)$$

This gives the final form for calculating the slope of the data points using a numerical least-squares fit:

$$m = \frac{\sum t_i(y_i - y_{-i})}{2 \sum t_i^2} \quad , \quad (4.18)$$

where  $i = 1, 2, 3, \dots, \text{INT} [(n/2) + 1]$  .

The effectiveness of this algorithm may be shown by illustration. Consider the data points shown in Fig. 11, which have the functional form  $y = 2x + 5$ , when  $x$  is a variable.

We first translate the horizontal reference frame so that the center data point becomes the new origin:  $y_0$  at  $x = 4$ ,  $y_1$  at  $x = 5$ ,  $y_{-1}$  at  $x = 3$ , etc.

Using these values in Eq. (4.18) gives:

$$\begin{aligned} m &= \frac{(1)(15 - 11) + (2)(17 - 9) + (3)(19 - 7)}{1^2 + 2^2 + 3^2} \quad , \\ &= \frac{4 + 16 + 36}{28} = 2 \quad . \\ b &= \frac{7 + 9 + 11 + 13 + 15 + 17 + 19}{7} = 13 \quad . \end{aligned}$$

The value for  $b$  is that of  $y_0$  in the numerical computation or  $y_4$  in the physical model. The value of the  $y$  intercept can easily be found from  $b$ . However, for the flow measurement, only the slope of the density compensated level is required, and  $b$  may be neglected.

### 4.3 PROGRAM DEVELOPMENT

The implementation of Eq. (2.28) to calculate the average flow rate is described in Sect. 5. All channels are similar, so only one channel will be discussed.

An overall flow diagram of the software architecture is shown in Fig. 12. The program is resident in EEPROM, and the system begins operation from power-up when power is applied. The system is initialized and the P-FORTH card waits for an interrupt. There are two possible interrupt conditions in this system: an interrupt is generated from a front-panel reset button or from a hardware timer. The front-panel reset button is used to reset an individual channel without affecting the other channels, and the hardware timer is used to generate a 6-s sample interval signal.

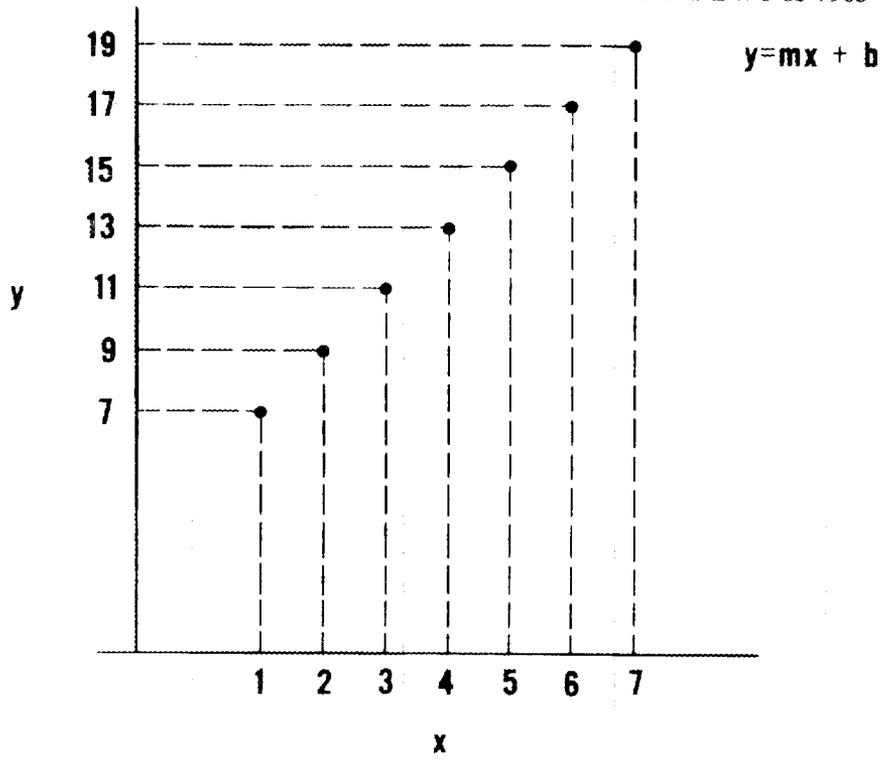


Fig. 11. Data points for function  $y = 2x + 5$ .

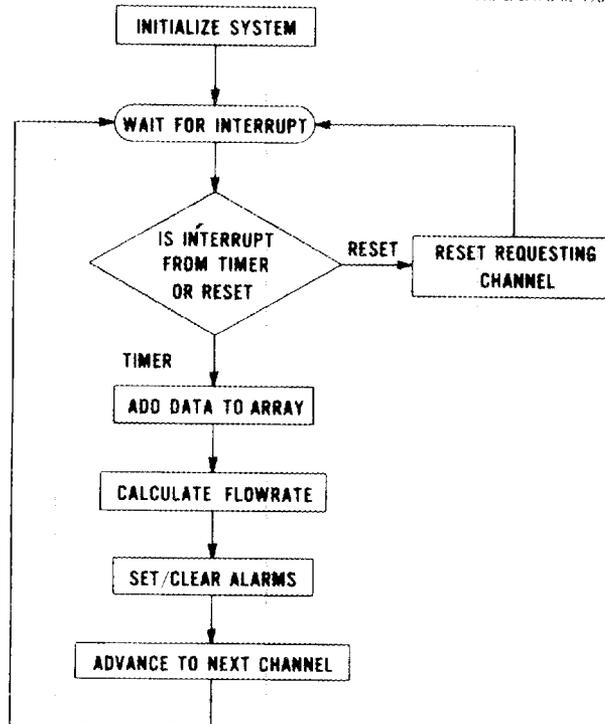


Fig. 12. Flowmeter software block diagram.

When the P-FORTH microprocessor receives an interrupt, it interrogates the system to determine the source of the interrupt. The microprocessor will then call the lower TIMER subroutine or the lower RESET subroutine.

The TIMER subroutine is executed by the FORTH word TIMER and the RESET subroutine by the word RESET. This is an example of the self-documenting nature of a well written FORTH program. These words are easily remembered and their functions are self-explanatory. A complete listing of the FORTH application VOCABULARY and a description of the GLOSSARY are included in the appendixes.

#### 4.4 SYSTEM INITIALIZATION

Upon power-up, a reset vector will execute a system initialization as shown in Fig. 13 by the FORTH word RESTART. The word RESTART will clear the data stack and the four data arrays. It also resets the channel counter to the first channel and clears each display. RESTART initializes the hardware timers of the Model 6522 PIA-integrated circuit, sets a vector to the interrupt service routine, and enables the interrupts. After RESTART has initialized the system, the program goes to the FORTH monitor to await an interrupt condition.

#### 4.5 DATA ARRAY

The primary element of each channel is the data array in RAM. Each channel is allocated 250 bytes for storage of the vessel volume samples: 202 bytes are set aside for 101 volume data points, and the remainder is used for holding channel information such as flow, density, and array size. A map of a typical array is shown in Fig. 14.

The array can contain up to 101 volume samples. The data counter register contains the number of data points in the array and is used by the program to calculate the least-squares fit of the data points.

#### 4.6 RESET SERVICE ROUTINE

If the interrupt source is from a front-panel reset, the microprocessor will reset that channel if it has finished the TIMER service routine. The RESET service routine is shown in Fig. 15. The reset/analog board latches a channel reset command and generates an interrupt signal. The microprocessor then reads the requesting channel from the reset/analog board.

After determining the requesting channel, the microprocessor uses the FORTH word RESET to execute a channel reset. The command RESET expects the number of the requesting channel to be on the data stack, and RESET

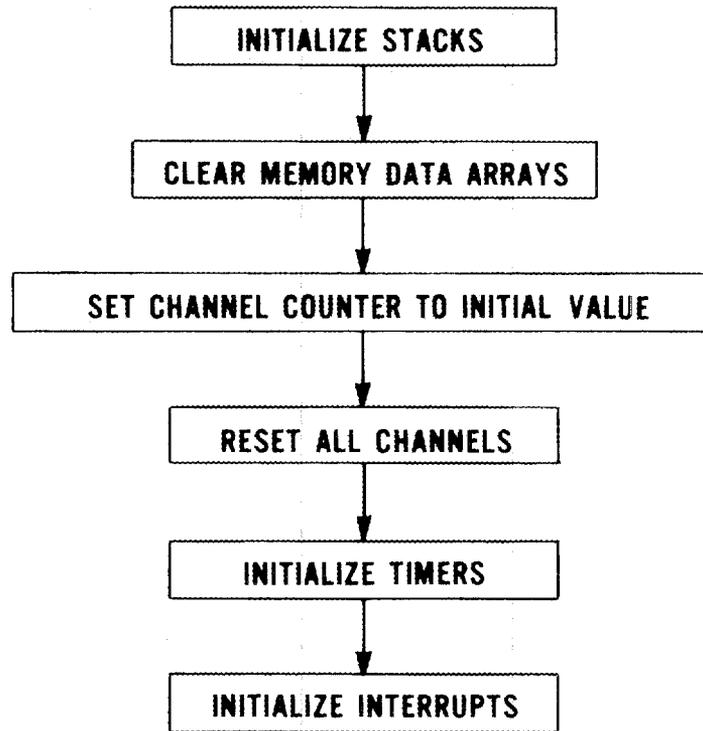


Fig. 13. System initialization flow diagram.

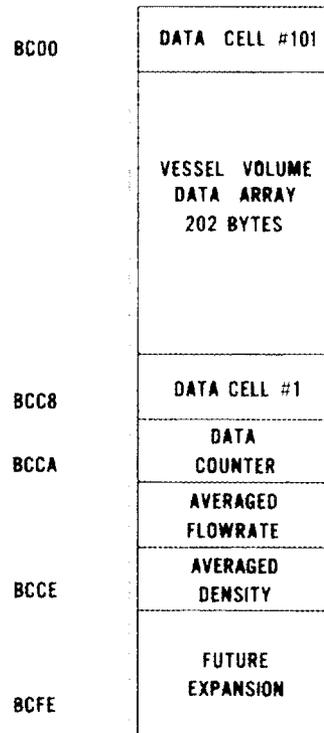


Fig. 14. Memory map of Channel One data array.

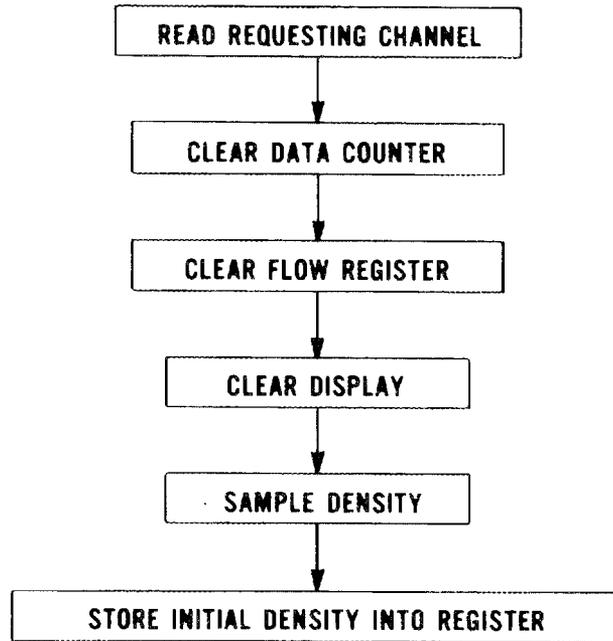


Fig. 15. RESET interrupt service flow diagram.

will clear the data counter, flow register, density register, and front-panel display of the requesting channel. It then takes an initial density sample and stores it into the density register, then RESET returns to the monitor.

The TIMER service routine has higher priority, so the microprocessor will not acknowledge a RESET interrupt until it has completed the TIMER service.

#### 4.7 TIMER SERVICE ROUTINE

If an interrupt is generated from the hardware timer, the program executes the TIMER service routine, as shown in Fig. 16. A timer generated interrupt is issued at 1.5-s intervals. The program executes the TIMER service routine for the active channel, advances to the next channel, and waits for the next timer interrupt. Since there are four channels, any channel is updated every 6 s. All channels use the same service routine, with incoming and outgoing data vectored to the appropriate channel locations in the memory array.

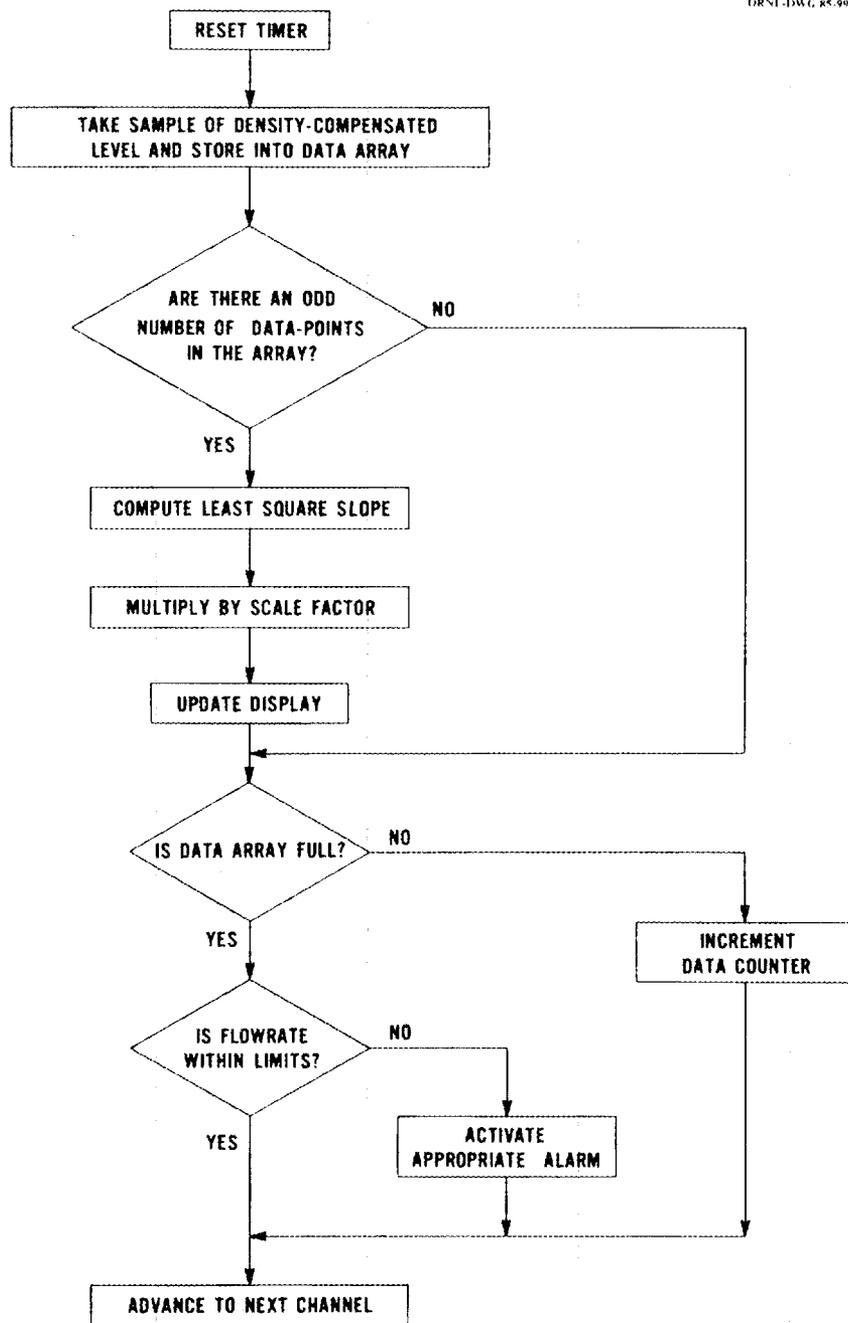


Fig. 16. TIMER interrupt service flow diagram.

The first action of TIMER is to reset the hardware timer on the 6522 PIA. A sample of the density compensated level is taken from the containment vessel. The vessel level is determined by making an analog-to-digital conversion from the differential pressure transmitter that is connected to the bubbler system level tube. The level is sampled 100 times and averaged to improve measurement precision and to reduce the effects of bubbles and artifacts on the measurement.

The density sample is taken similarly, except that only 20 samples are averaged. This density value is averaged with the previous density value and stored in the data array for averaging with the next sample. The recursion uses equal parts of the old and new densities.

The average level is multiplied by 5 and divided by 5 times the density plus a constant, which is the density compensated level or the Q term from Eq. (2.27).

The memory array is shifted by one cell to allow room for the newly acquired sample. Data cell #1 moves to cell #2, cell #2 moves to cell #3, and so on. The most recently acquired sample is written into cell #1, and the data counter is checked to see if there is an odd number of samples in the array. If the number of data samples in the array is odd, the program computes the average flow rate and updates the display. If there is an even number of data samples, then a least-squares fit cannot be done on the array.

The program computes the average flow rate by calculating the best-fit slope of the points in the array. The slope is multiplied by a scaling factor and averaged with the previous flow. The recursion is such that a change of 0.1 mL/min is seen on the display if a flow increase of 0.1 mL/min is maintained for three consecutive samples. This check prevents slight perturbations in the data array from affecting the displayed flow rate. It is desirable to stabilize the display so as not to create ambiguous readings that might mislead a control room operator. The slope of the array is found using the FORTH word LSFIT. LSFIT takes the data counter value, indexes it into the center of the array, and calculates the slope  $\Delta Q/\Delta t$  by the previously discussed least-squares fit algorithm.

The averaged flow rate is rounded to the nearest 0.1 mL/min and written to the appropriate digital front-panel display.

The data counter is checked to see if the array contains 101 data points, which would indicate that the array is full. If the array is full, the flow rate is compared with upper and lower flow limits to ascertain if the flow is within the acceptable range. If the flow rate is lower than 4.0 mL/min, an alarm is activated that closes a solid-state relay. The relay closure illuminates an UNDERFLOW warning lamp for the corresponding channel. A flow above 14.0 mL/min will similarly cause an OVERFLOW to illuminate. No action is taken if the flow is within tolerance.

The flow rate check is not performed unless the data array is complete. This delay is to allow startup transients to decay before the alarms are enabled. If the data counter shows the array has less than 101 data points, the data counter is incremented.

The final action by the TIMER service routine is to increment the channel counter, which advances the program to the next channel and controls the sequential multiplexing of the four channels. The active channel is stored in RAM in the variable CHAN.

After the interrupt service routines have been completed, the program returns to the FORTH monitor. The monitor continuously scans the input for the next interrupt, whereupon the sequence is repeated.

## 5. EXPERIMENTAL RESULTS

A series of tests were conducted to verify the performance of the flow monitor. Tests were run that proved the goodness of fit of the least-squares algorithm and established the actual operational characteristics of the device under field conditions.

### 5.1 ANALOG INPUT FILTER

The use of the 125-Hz analog filter in the reset/analog board was justified by comparing data acquired with the filter and without. The analog filter is located in the input signal line in series with the differential pressure-to-current converter and the ADC. Data taken without the filter are shown in the upper trace of Fig. 17, and the reduction of artifacts is readily apparent in the lower trace.

### 5.2 VERIFICATION OF MODEL

A test vessel was constructed to simulate the actual containment vessel and had an inside diameter of 6 13/32 in. A bubbler system shown in Fig. 18 was installed with the density separation, D, adjusted to 10 in. The maximum pressure capacity of the differential pressure-to-current converter,  $P_L$ , was 32 in. of water, while  $P_D$  was 20 in. of water. On the bottom of the test vessel was an outlet with a fluid rotameter and needle valve that measured and adjusted the outgoing flow.

The tank was filled with water, and the flow was collected and measured in the graduated cylinder. The collected water volume was compared with the flow monitor readings.

### 5.3 SCALE FACTOR

The scaling factor was tested by comparing the theoretically derived value with an experimentally determined value. Since the scale factor is  $K = (4/5) \pi R^2 H_L$  from Eq. (2.26), by inserting the appropriate parameters from the test vessel, we get

$$K = \left(\frac{4\pi}{5}\right)(32 \text{ in.}) \left(\frac{6.41}{2} \text{ in.}\right)^2 \left(2.54 \frac{\text{cm}}{\text{in.}}\right)^3 ,$$

$$K = 13538 \text{ mL} .$$

The scale factor was determined experimentally by sampling the vessel volume with the ADC and recording the result. The needle valve of the vessel was opened and 100 mL was collected in the graduated cylinder. The needle valve was then closed and the vessel volume was measured. To determine the scale factor, the difference between the volumes is

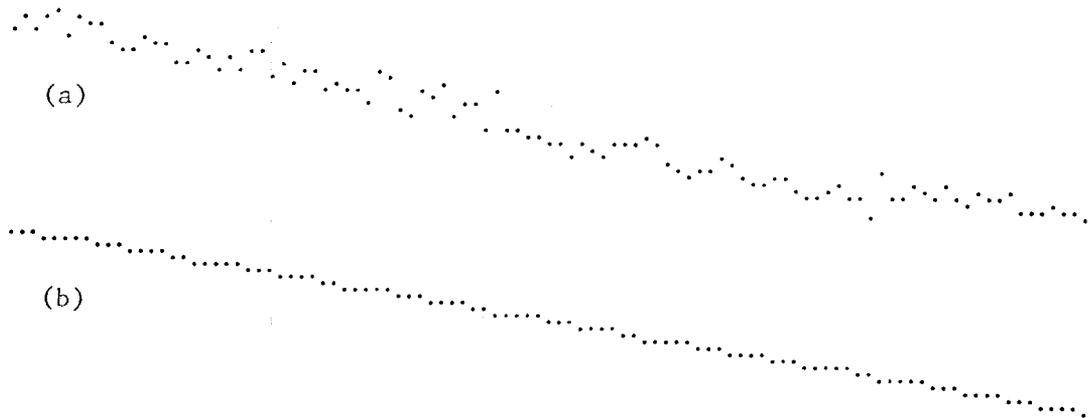


Fig. 17. Effect of analog filter on input signal. (a) Input signal without 125-Hz low-pass filter; (b) input signal with filter.

divided into the volume collected. This procedure was used when the vessel was nearly full, half full, and nearly empty. The results were averaged to get an experimental scale factor,

$$K = 13898 \text{ mL}$$

The theoretical scale factor differs by Error  
 $= [(13898 - 13538)/13898] \times 100 = 2.6\%$ , which is within acceptable limits of the measurement precision.

#### 5.4 LEAST-SQUARES ALGORITHM

The accuracy of the least-squares-fit algorithm was tested by loading the array with a predetermined sequence of numbers, which was calculated from the mathematical slope that would correspond to an arbitrary flow rate. The least-squares routine was run, and the results of the least-squares fit were compared with the mathematical flow rates. The test was run for 60 slopes within the range of anticipated flow rates, and the results of this experiment showed the least-squares-fit algorithm to be within 0.2 mL of the mathematical flow. All deviations from the theoretical flow were negative.

#### 5.5 STATISTICAL SAMPLING

The sampling technique used in the flow monitor reduces sampling error through averaging by making many individual analog-to-digital conversions and averaging the results. A frequency-of-occurrence

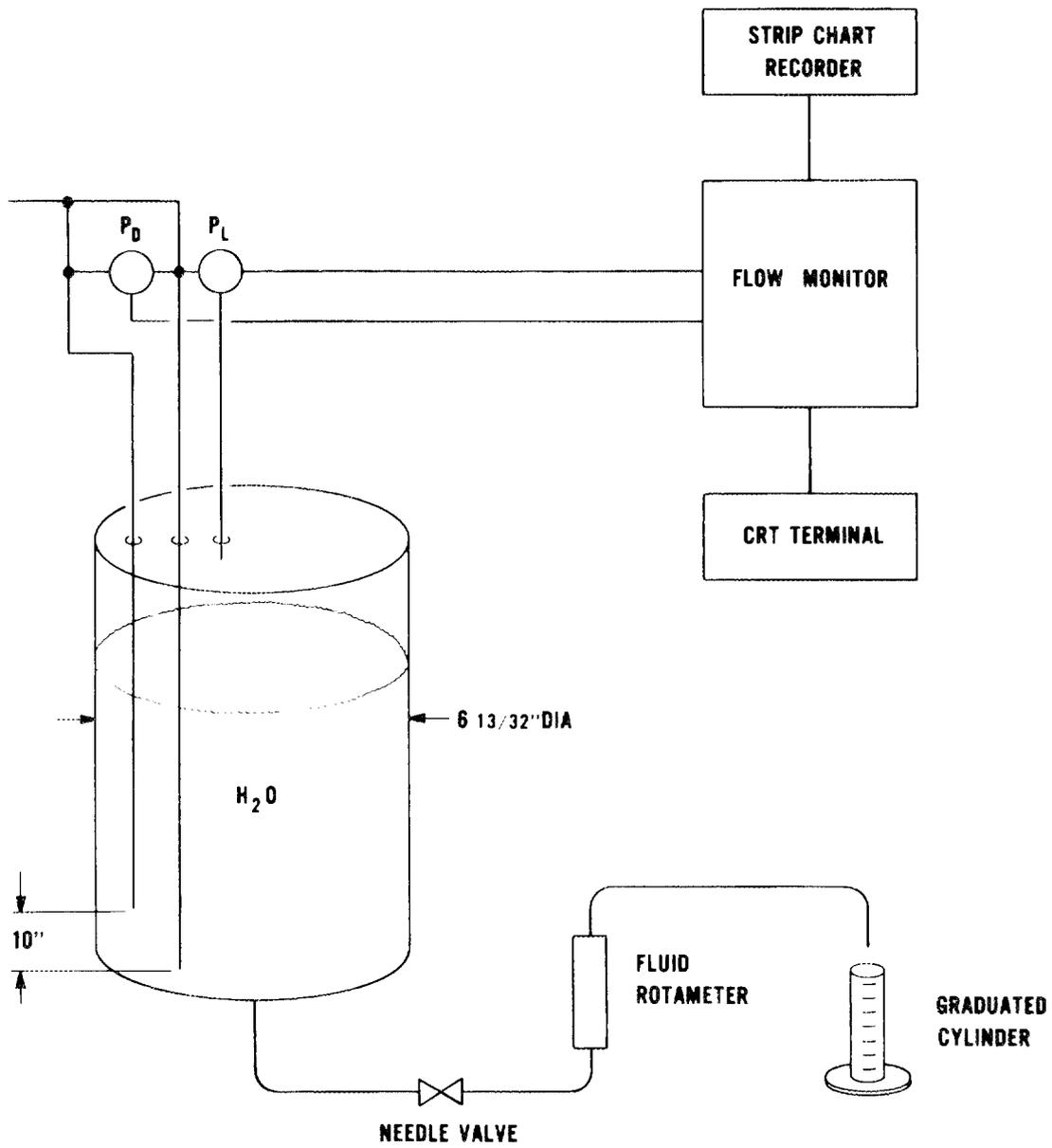


Fig. 18. Diagram of experimental apparatus.

distribution graph for a density measurement is shown in Fig. 19. The vertical axis is the number of occurrences, and the horizontal axis is a normalized digital value. The distribution on the left is from a single-sample process, and the distribution on the right is similar, except that only 20 single samples were averaged. A statistical comparison of the two distributions is shown below.

	<u>Mean</u>	<u>Standard deviation</u>
Single sample	2.7	0.84
Averaged sample	2.6	0.65

Through repeated tests, the standard deviation of the sampling distribution proved to be reduced through averaging. No correlation could be established between the number of samples averaged and the amount of reduction. This lack of agreement is probably an indication that the sampling process is not random. The average standard deviation for a single-sample measurement was 1.1, while the average deviation for the multisample measurement was 0.8.

#### 5.6 OPERATIONAL TESTS

Operational tests were run on the measurement instrument to verify accuracy under field conditions. The needle valve was set to an arbitrary flow rate within the anticipated normal operating range, water output was collected in a graduated cylinder for 10 min, the cylinder was removed, and the collected volume was recorded. This collected volume was then compared with the displayed flow in the test instrument. This procedure was done with the data array initially empty and after the data array was complete. The collected volume was compared with the calculated flow rate of a least-squares-fit slope and with an endpoint-derived average slope. The results of this group of tests are shown in Fig. 20. The three traces show the deviations of the flow rate monitor from the actual flow rate. (The actual flow rate is defined as the collected volume divided by the collection period.)

The upper trace of Fig. 20 shows the deviations in milliliters per minute of the endpoint-derived slope, which was calculated as the difference between the initial and the final array entry. The middle trace is the deviation of the least-squares slope from initial conditions. The lower trace shows the deviation of the least-squares slope under steady-state conditions that would be the normal operating mode of the measurement instrument. The standard deviations of the conditions are shown below.

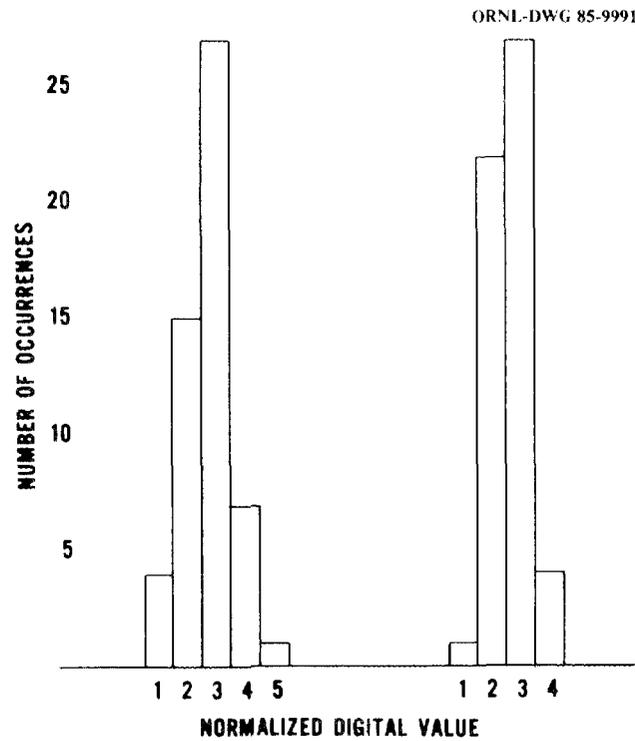


Fig. 19. Frequency distribution histogram for data sampling process.

<u>Method</u>	<u>Standard deviation</u>
Endpoint transient	0.586
Least-squares transient	0.18
Least-squares steady-state	0.098

The standard deviation of the steady-state, least-squares method is seen to be nearly one-sixth that of the endpoint measurement, differing from the expected reduction of one-tenth predicted in Sect. 2. This difference probably is due to less than the expected deviation of the endpoint measurement. It is likely that the pressure sensor accuracy was better than the rated 0.25%.

#### 5.7 TRANSIENT RESPONSE

The transient response of the monitor was approximately second order. The overshoot was variable with a maximum value of 200%. High overshoots occur when the array has few data points, and these quickly subside as more data are taken. The typical 5% settling time was 5 min.

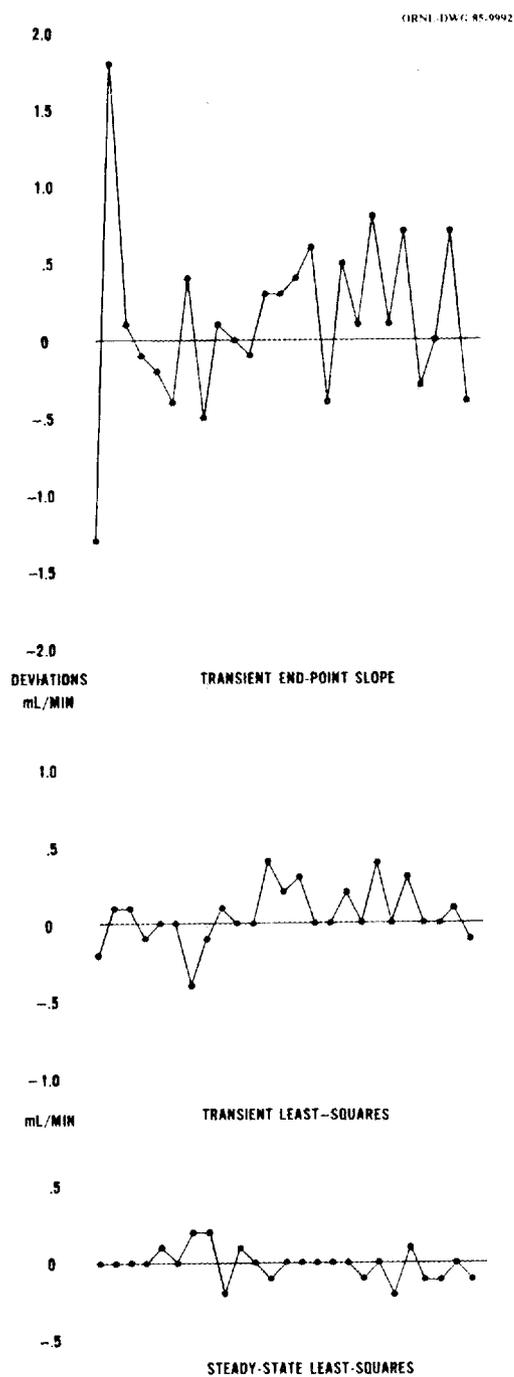


Fig. 20. Deviations of calculated flow from measured flow.

## 5.8 LONG-TERM STABILITY

A measure of the long-term stability of the instrument was made by closing the needle valve and recording the flow rate on a chart recorder. Any deviation from zero flow would be an inaccuracy of the system. The maximum deviation seen over an 18-h period was 0.4 mL/min.

## 6.0 SUMMARY AND CONCLUSIONS

A flow rate monitor was developed to measure the average flow of a variable-density fluid from a cylindrical holding tank. The flow rate monitor uses an ADC to sample the level and density of the fluid. A single-board STD bus computer normalizes these signals to a density compensated level. The density compensated level sample is stored in a data array in the computer memory. At 6-s intervals a new sample is taken and added to the array. As each new sample is added, the oldest sample is dropped. The array contains the history of the density compensated level for the most recent 10-min period. A linear regression is run on the array to determine the best fit of a straight-line slope of level versus time. The slope is defined as the average change of level in the holding tank, and this number is multiplied by a dimensional constant and averaged with the two preceding slopes. This number is displayed as the average flow rate from the holding tank. An alarm is activated if the flow rate exceeds prescribed limits of operation.

The intended function of the flow rate monitor is to provide supervisory monitoring of a chemical process, with deviations reported to a control room operator via an annunciator panel. The flow rate monitor could easily be adapted to provide process control input and allow automatic regulation of flow.

The monitor met or exceeded all required specifications (Sect. 1.2) and has been reliable in long-term tests conducted to date, although an improved sensor would increase the instrument accuracy. The pressure-to-current converter used for testing had an accuracy of 0.25%. Pressure-to-current converters with 0.1% accuracy are available at higher cost.

Future work might include checking the standard deviation of the array. It is possible to make a numerical calculation of the statistical characteristics of the array. The instrument could respond with an error condition if the standard deviation exceeded preset upper and lower limits, which could be established based on the flow range and sensor accuracy. Samples falling outside these limits could indicate a defective component in the measurement process.

Computational speed is the limiting factor for additional improvements because numerical statistics require large amounts of data crunching. Increasing the workload of the microprocessor must be paid for by increasing the clock speed, which becomes more feasible as faster and more powerful microprocessors become available.

## REFERENCES

1. Model GC-781 Moore Pneumatic to Electric Transducer, Data Sheet GC-781, Moore Products Co., Spring House, Pa., 1982.
2. Thornton C. Fry, Probability and Its Engineering Uses, 2d ed., Van Nostrand Co., Princeton, N.J., 1965, p. 342.
3. P-FORTH Users Manual, Peopleware Systems, Inc., Minneapolis, 1982.
4. RTI-1260 Users Manual, No. AC1563, Analog Devices, Inc., Norwood, Mass., 1981.
5. OPTO-22 Data Book, OPTO-22, Inc., Huntington Beach, Calif., 1983.
6. 7507 General Purpose I/O Interface Card Users Manual, Pro-Log Corp., Monterey, Calif., 1982.
7. 7602 Output Port Card Users Manual, Pro-Log Corp., Monterey, Calif., 1981.
8. 7904 Decoded Utility I/O Card Users Manual, Pro-Log Corp., Monterey, Calif., 1982.
9. E. D. Rather and C. H. Moore, FORTH High-Level Programming Technique on Microprocessors, FORTH, Inc., Manhattan Beach, Calif., 1976.

APPENDIX A

FORTH APPLICATION PROGRAM LISTING

## FORTH APPLICATION PROGRAM LISTING

2 VARIABLE CHAN

2 VARIABLE ISUM

0152H CONSTANT ALARM

01FFH CONSTANT RPORT

004DH CONSTANT IFR

0040 CONSTANT LOWER

0140 CONSTANT UPPER

CODE CONV

```

          PULA
          PULB          ; MOVE STACK TO ACCUMULATOR
010BH    STAB          ; SELECT MUX CHANNEL
          BEGIN
010D     LDAA
          MI            ; IS CONVERSION COMPLETE?
          NOT UNTIL    REPEAT UNTIL COMPLETE
010DH    LDAA          ; READ UPPER BYTE
010C     LDAB          ; READ LOWER BYTE
          PSHB
          PSHA          ; POP TO STACK
          NEXT JMP     ; RETURN TO FORTH
          END-CODE

```

CODE J

```

008EH    LDX          ; LOAD RETURN STACK ADDRESS INTO
          INDEX
          INX
          INX
          INX
          INX
          INX
          INX          ; ADD 6 TO LOCATE 3RD ITEM
C012H    JMP          ; MOVE TO STACK, RETURN TO FORTH
          END-CODE

```

```

SELECT          <BUILDS DOES> CHAN @
                2 * + @ ;

```

SELECT 'ARRAY

BCC8, BDC8, BEC8, BFC8,

```

SELECT    MASK
          FC, F3, CF, 3F,

SELECT    SCALOR
          14098, 14098, 14098, 14098,

:SHIFT    'ARRAY 200 - DUP 2 - 202 CMOVE ;

:COUNTER  'ARRAY 2 + ;

:FLOW     'ARRAY 4 + ;

:DENS     'ARRAY 6 + ;

:ROUND    10 /MOD SWAP 4 > IF 1+ THEN ;

:MEMCLR   BBC8H 0437H ERASE ;

:LCON     CHAN @ 2 * CONV ;

:DCON     CHAN @ 2 * 1+ CONV ;

:S/       U/ SWAP DROP ;

:SCALE    SCALOR 1000 * / ;

:DENSITY  00 00 20 0 DO DCON 0 D+ LOOP
          2 S/ 0 DENS @ 0 D+
          2 S/ DUP DENS ! ROUND ;

:LEVEL    00 00 100 0 DO LCON 0 D+ LOOP 100 S/ ;

:ADCON    LEVEL 5 * 10,000 DENSITY 5 *
          12288 + */ ;

:LSFIT    'ARRAY OVER 1 - - >R
          0 ISUM ! 1+ 2 /
          0 0 ROT 1 DO
          I I * ISUM +!
          J I 2 * + @
          J I 2 * - @ -
          100 I * M* D+ LOOP
          R> DROP DABS ISUM @ S/ 2 / ;

:DISPLAY  DUP 1000 < IF 100 /MOD
          256 * SWAP 10 /MOD 16 * + +
          ELSE -1 THEN
          CHAN @ 2 * 256 + ! ;

```

```

:LIM-BIN      1 CHAN @ - DUP IF 0 DO 4 * LOOP THEN;

:WITHIN      FLOW @ DUP LOWER < SWAP UPPER >

              2 * + ;
:AVG         FLOW @ SWAP OVER - 3 / +
              DUP FLOW ! ;

:?LIMIT      WITHIN LIM-BIN * ALARM C@
              MASK AND OR ALARM C! ;

:FLOWRATE    COUNTER @ LSFIT SCALE
              AVG ROUND DISPLAY ;

:TIMER       0049H C! SHIFT ADCON 'ARRAY !
              COUNTER @ 2 MOD IF FLOWRATE THEN
              COUNTER @ 65H = IF ?LIMIT ELSE 1 COUNTER +!
              THEN CHAN @ 3 = IF 0 CHAN !
              ELSE 1 CHAN +! THEN ;

:RESET       CHAN @ SWAP CHAN ! 0 COUNTER !
              0 FLOW ! 0 DISPLAY
              DCON 10 * DENS !
              CHAN ! ;

:CHAN?       40H @ DROP RPORT C@
              0 RPORT C! ;

:RESET-ALL   4 0 DO I RESET LOOP ;

:INTR        OFFIRQ IFR C@ 20H AND IF
              TIMER THEN IFR C@ 01 AND IF
              CHAN? RESET 01 IFR C! THEN
              ONIRQ QUIT ;

CODE INTLINK
BA6D #LDS,      ; clear data stack
NEXT # LDX,
PSHX,          ; load NEXT onto data stack
1 INTR
2 - #LDX,      ; put address of INTR in index
               register
8C STX,        ; move INTR address to I register
RTS,           ; return from subroutine
END-CODE       ; return to FORTH and execute INTR

:RESTART     SP! SETIRQ INTLINK 0
MEMCLR        0 CHAN ! RESET-ALL
EO 4B C!
BF 42 C!

```

```

FF 43 C!
627F 44 !
A1 4E C!
16 48 C!
00 49 C!
02 4C C!
CHAN? DROP DECIMAL ONIRQ ;

```

AUTO RESTART

#### FORTH APPLICATION GLOSSARY

- CHAN - 2-Byte variable that contains the currently active channel.
- ISUM - 2-Byte variable used in LSFIT to accumulate the sum of time periods squared.
- ALARM - 0152 HEX constant of the ALARM card I/O port.
- RPORT - 01FF HEX constant of the RESET card I/O port.
- IFR - 04D HEX constant of the 6522 interrupt flag register.
- CONV - (n--n) Assembly language routine to perform an analog-to-digital conversion. Expects the channel number to be on the stack and leaves the results on the stack.
- J - Used during a DO loop to copy the third item on the return stack to the parameter stack.
- SELECT - Defining word that allows creation of memory arrays. When words created by SELECT are executed, elements in the array of the active channel are readily available.
- ARRAY - (--n) An array created by SELECT that holds the addresses of data cell #1 for each channel. When 'ARRAY is executed, it reads the active channel from CHAN and leaves the corresponding channel address on the stack.
- MASK - (--n) An array created by SELECT that contains a bit pattern to mask out the alarm functions of inactive channels. Used in ?LIMIT.
- SCALOR - (--n) An array created by SELECT that returns the scaling factor for the active channel. Used in FLOWRATE.

- SHIFT - Moves the elements in the data array down 2 bytes. Cell #1 moves to cell #2, cell #2 moves to #3, etc. Begins with cell #100 and works back to cell #1. Cell #101 is eliminated.
- COUNTER - Register in data array that holds the size of the array.
- FLOW - Register in data array that contains the current flow rate of that channel.
- DENS - Register in data array that holds the channel density.
- MEMCLR - Used in initialization to clear all data arrays.
- LCON - (--n) Performs an analog-to-digital conversion for the level of the active channel and leaves result on stack.
- DCON - (--n) Same as above except does conversion of density of active channel.
- S/ - (dn--n) Divides a double-precision number by a single-precision quotient on the stack.
- SCALE - (n--n) Multiplies number on stack by the scale factor of active channel and leaves results on stack.
- ROUND - (n--n) Takes number on stack, rounds off to nearest tenth, and leaves results on stack.
- DENSITY - (--n) Makes 20 density measurements of active channel. Takes the average of the 20 measurements and then averages with the previous sampling period density.
- LEVEL - (--n) Makes 100 level measurements and leaves the average on the stack.
- ADCON - (--n) Takes 5 times DENSITY and adds 12288, then divides into 5 times LEVEL. This is the value stored in the data array as the density compensated level.
- LSFIT - (n--n) Takes the array size on the stack and does a least-squares fit on the data array. Leaves the slope on the stack.
- AVG - (n--n) Takes 1/3 of flow from stack and adds to 2/3 of value in FLOW. Places new value into FLOW and leaves a copy on stack.

- DISPLAY - (n--) Takes number from stack, performs a decimal-to-hex conversion, and displays value on active channel display.
- LIM-BIN - (--n) Generates a binary limit for the active channel and is used to vector the correct alarm location in ?LIMIT.
- WITHIN - (n--f) Takes a number from stack and checks to see if the number is within the fixed upper and lower limits. Leaves a flag 0, 1, or 2 to indicate normal, under, or over condition.
- ?LIMIT - (n--) Takes flow rate from stack and compares it to upper/lower limits. Activates the appropriate alarm if necessary.
- FLOWRATE - (--n) Computes the flow rate of the active channel using LSFIT, SCALE, AVG, and ROUND. Updates display with flow rate and leaves copy on stack.
- TIMER - TIMER interrupt service routine that resets hardware timer, takes density compensated level sample, and calculates flow rate.
- RESET - (n--) RESET interrupt service routine. Clears data counter and flow register and sets density register to initial value.
- CHAN? - (--n) Returns channel number when a front-panel reset button has been depressed.
- RESET-ALL- Used in initialization to RESET all channels.
- INTR - Interrupts service routine that polls the interrupt flag register to determine if a RESET or TIMER interrupt has occurred.
- INTLINK - Assembly language word that vectors to INTR upon receipt of an interrupt.
- RESTART - Initialization word that clears the system and sets all operating parameters to their initial values. Executes automatically from power-up to initialize system and begin execution of flow rate program.



APPENDIX B

PORTIONS OF THE FIG-\* FORTH GLOSSARY USED IN THE P-FORTH MODEL

The FIG-\* FORTH Glossary was developed by the  
Forth Interest Group, San Carlos, California.

## A. P-FORTH GLOSSARY

Explanation of Glossary

In this glossary, word names are listed in two subsequences:

1. The main subsequence is for all those word names that include letters or numbers. Within this group, word names are in order of their alphanumeric content.
2. The other subsequence is for those names that do not contain letters or numbers. The nonalphanumeric subsequence comes first in the Glossary.

The first line of each entry shows a symbolic description of the effect on the parameter stack. The symbols indicate the order in which input parameters have been placed on the stack. Three dashes "---" indicate the execution point; any parameters left on the stack are listed. In this notation, the top of the stack is to the right.

The symbols include:

```

addr  memory address
b     8-bit byte (i.e., high 8 bits zero)
c     7-bit ASCII character (high 9 bits zero)
d     32-bit signed double integer, most significant
      portion with sign on top of stack
f     Boolean flag. 0=false, non-zero=true
ff    Boolean false flag - 0
n     16-bit signed integer number
u     16-bit unsigned integer
tf    Boolean true flag-non-zero
tp    true part
fp    false part

```

The capital letters on the right show definition characteristics:

```

C     May be used only within a colon definition. A digit indicates
      number of memory addresses used, if other than one.
E     Intended for execution only.
I     Indicates that the word is immediate and will execute even when
      compiling, unless special action is taken.
U     A user variable.

```

Unless otherwise noted, all references to numbers are for 16-bit signed integers. The high byte of a number is on top of the stack, with the sign in the left-most bit. For 32-bit signed double numbers, the most significant part (with the sign) is on top.

All arithmetic is implicitly 16-bit signed integer math, with error and underflow indication unspecified.



:		I,E
	Used in the form called a colon-definition: : cccc ... ; Creates a dictionary entry defining cccc as equivalent to the following sequence of FORTH word definitions '...' until the next ';' or ';CODE'. The compiling process is done by the text interpreter as long as STATE is non-zero.	
;	---	I,C
	Terminate a colon-definition and stop further compilation.	
<	n1 n2 --- f	
	Leave a true flag if n1 is less than n2; otherwise leave a false flag.	
=	n1 n2 --- f	
	Leave a true flag if n1 = n2; otherwise leave a false flag.	
>	n1 n2 --- f	
	Leave a true flag if n1 is greater than n2; otherwise leave a false flag.	
@	addr --- n	
	Leave the 16-bit contents of address. Pronounced "fetch."	
0=	n --- f	
	Leave a true flag if the number is equal to zero; otherwise leave a false flag.	
1+	n1 --- n2	
	Increment n1 by 1.	
2+	n1 --- n2	
	Leave n1 incremented by 2.	
ABS	n --- u	
	Leave the absolute value of n as u.	
AND	n1 n2 --- n3	
	Leave the bitwise logical AND of n1 and n2 as n3.	

AUTO

---

Used in the form: AUTO <Name> to initialize a pointer in EEROM. On power-up if target switch is on, <Name> is executed immediately.

BEGIN--- addr n (compiling)

I

Occurs in a colon-definition in the form:

```
BEGIN ... UNTIL
BEGIN ... AGAIN
BEGIN ... WHILE ... REPEAT
```

At run time, BEGIN marks the start of a sequence that may be executed repetitively. It serves as a return point from the corresponding UNTIL, AGAIN, or REPEAT. When executing UNTIL, a return to BEGIN will occur if the top of the stack is false; for AGAIN and REPEAT, a return to BEGIN always occurs.

At compile time, BEGIN leaves its return address and n for compiler error checking.

<BUILDS

C

Used within a colon-definition: : cccc <BUILDS ...  
DOES> ... ;

Each time cccc is executed, <BUILDS defines a new word with a high-level execution procedure. Executing cccc in the form: cccc nnnn uses <BUILDS to create a dictionary entry for nnnn with a call to the DOES> part for nnnn. When nnnn is later executed, it has the address of its parameter area on the stack and executes the words after DOES> in cccc. <BUILDS and DOES> allow run-time procedures to be written in high-level rather than assembler code (as required by ;CODE).

C!

b addr ---

Store 8 bits at address. Pronounced "c-store."

C,

b ---

Store 8 bits of b in the next available dictionary byte, advancing the dictionary pointer.

C@

addr --- b

Leave the 8-bit contents of memory address. Pronounced "c-fetch."



CSP --- addr U

A user variable temporarily storing the stack pointer position, for compilation error checking.

CURRENT --- addr U

A user variable containing a pointer to the vocabulary to which new definitions are compiled.

D+ d1 d2 --- dsum

Leave the double number sum of two double numbers.

DABS d --- ud

Leave the absolute value ud of a double number.

(DO) C

The run-time procedure compiled by DO that removes the loop control parameters to the return stack. See DO.

DO n1 n2 --- (execute)  
addr n --- (compile) I,C2

Occurs in a colon-definition in form:

DO ... LOOP

At run time, DO begins a sequence with repetitive execution controlled by a loop limit n1 and an index with initial value n2. DO removes these from the stack. Upon reaching LOOP, the index is incremented by one. Until the new index equals or exceeds the limit, execution loops back to just after DO; otherwise the loop parameters are discarded and execution continues ahead. Both n1 and n2 are determined at run time and may be the result of other operations. Within a loop, 'I' will copy the current value of the index to the stack. See I, LOOP, LEAVE.

DOES>

A word that defines the run-time action within a high-level defining word. DOES> alters the code field and first parameter of the new word to execute the sequence of compiled word addresses following DOES>. Used in combination with <BUILDS. When the DOES> part executes, it begins with the address of the first parameter of the new word on the stack. This allows interpretation using this area or its contents. Typical uses include the FORTH assembler, multidimensional arrays, and compiler generation.



Occurs in a colon-definition in form:

```
IF (tp) ... THEN
IF (tp) ... ELSE (fp) ... THEN
```

At run time, IF selects execution based on a Boolean flag. If f is true (non-zero), execution continues through the true part. If f is false (zero), execution skips till just after ELSE to execute the false part. After either part, execution resumes after THEN. ELSE and its false part are optional; if missing, false execution skips to just after THEN.

(LOOP) C2

The run-time procedure compiled by LOOP that increments the loop index and tests for loop completion. SEE LOOP.

LOOP addr n --- (compiling) I,C2

Occurs in a colon-definition in the form: DO ... LOOP

At run time, LOOP selectively controls branching back to the corresponding DO based on the loop index and limit. The loop index is incremented by one and compared to the limit. The branch back to DO occurs until the index equals or exceeds the limit; at that time, the parameters are discarded and execution continues.

M\* n1 n2 --- d

A mixed magnitude math operation that leaves the double-number signed product of two signed numbers.

/MOD n1 n2 --- rem quot

Leaves the remainder and signed quotient of n1/n2. The remainder has the sign of the dividend.

OFFIRQ ---

Disables interrupts.

ONIRQ ---

Enables interrupts.

OR n1 n2 --- n3

Leaves the bit-wise logical OR of two 16-bit values.

OVER n1 n2 --- n1 n2 n1



U\*                    u1 u2 --- ud

Leave the unsigned double-number product of two unsigned numbers.

U/                    ud u1 --- u2 u3

Leave the unsigned remainder u2 and unsigned quotient u3 from the unsigned double dividend ud and unsigned divisor u1.

UNTIL                    f --- (run time)  
                          addr n --- (compile)                    I,C2

Occurs within a colon-definition in the form BEGIN ... UNTIL

At run time, UNTIL controls the conditional branch back to the corresponding BEGIN. If f is false, execution returns to just after BEGIN; if true, execution continues.

VARIABLE                    n ---

A defining word used in the form n VARIABLE <name> to create a dictionary entry for <name> and assign n bytes for storage in P-FORTH's variable storage area in RAM, which starts at location \$BA6F. The application must initialize the stored value. When <name> is later executed, it will place the address of the first byte of the assigned storage on the stack. Note that this differs from versions of FORTH that build their dictionary in RAM.

## B. DESCRIPTION OF FORTH ASSEMBLER

Introduction. P-FORTH supplies a FORTH-type assembler that supports user macros, literal values expressed in any numeric base, expressions using any resident computation capability, and nested control structures without labels and with error control.

This assembler is used to create execution procedures that would be time inefficient if written as colon-definitions. Functions may be written first in high level, tested, and recoded into assembly with a minimum of restructuring.

Using the Assembler. Invoking the assembler causes CONTEXT to be switched to the ASSEMBLER vocabulary. Each word in the input stream received from the CRT terminal will be matched according to the FORTH practice of searching CONTEXT first, then CURRENT.

The ASSEMBLER words in a CODE definition specify operands, address modes, and op-codes. At the conclusion of a CODE definition, a final error check verifies correct completion by "unsmudging" the definition's name, making it available for dictionary searches.

Run Time, Assembly Time. One must be careful to understand at what time a particular word definition executes. During assembly, each assembler word interpreted executes. Its function at that instant is called 'assembling' or 'assembly time.' This function may involve op-code generation, address calculation, mode selection, etc.

The later execution of the generated code is called 'run time.' This distinction is particularly important with the conditionals. At assembly time each such word (i.e., IF, UNTIL, BEGIN, etc.) itself 'runs' to produce machine code, which will later execute at what is labeled 'run time' when its named code definition is used.

Op Codes. The ASSEMBLER vocabulary includes a dictionary entry for each Model 6801 op-code (BSR and SWI are not included). These entries end in ",". The significance of this is:

1. The comma shows the conclusion of a logical grouping that would be one line of classical assembly source code.
2. "," compiles into the dictionary; thus a comma implies the point at which code is generated.
3. The "," distinguishes op-codes from possible hex numbers (e.g., ADDA and ADDB).

NEXT. FORTH executes user word definitions under control of the address interpreter, named NEXT. This short code routine moves execution from one definition to the next. At the end of code

definition, the user must return control to NEXT or else to code, which returns to NEXT. NEXT is a constant that specifies the machine address of FORTH's address interpreter. It is the operand for JMP,. As JMP, executes, it assembles a machine code jump to the address of NEXT from the assembly time stack value.

PUSHBA is the other location to which a JMP may be made. PUSHBA will push the two accumulators on the data stack and continue to NEXT.

Security. Numerous tests are made within the ASSEMBLER for user errors:

1. All parameters used in CODE definitions must be removed.
2. Conditionals must be properly nested and paired.
3. Address modes and operands must be allowable for the op-codes.

These tests are accomplished by checking the stack position (in CSP) at the creation of the definition name and comparing it with the position at END-CODE. The legality of address modes and operands is ensured by means of a bit mask associated with each operand.

Remember that if an error occurs during assembly, END-CODE never executes. The result is that the "smudged" condition of the definition name remains, and it will not be found during dictionary searches.

The user should be aware that one error not trapped is the referencing of a definition in the wrong vocabulary,

NOT of ASSEMBLER when you want  
NOT of FORTH .

## C. EXPLANATION OF ASSEMBLER GLOSSARY

The first line of each entry shows a symbolic description of the action of the procedure on the parameter stack. The symbols indicate the order in which input parameters have been placed in the stack. Three dashes "---" indicate the execution point; any parameters left on the stack are listed. In this notation, the top of the stack is to the right.

The symbols include:

addr	memory address
b	8-bit byte (i.e., high 8 bits zero)
f	Boolean flag. 0=false, non-zero=true
ff	Boolean false flag - 0
n	16-bit signed integer number
u	16-bit unsigned integer
tf	Boolean true flag - non-zero
cc	condition specifier.

In addition to the entries in this glossary, the ASSEMBLER includes one word for each Model 6801 mnemonic in the form:

ABX, where ABX is a standard 6801 mnemonic

The , suggests that code is compiled into the dictionary at this point.

Two mnemonics are not implemented:

SWI Since this is used by the system.

BSR Because the offset to a subroutine is not under the usual control and subroutines are seldom used.

Assembler Glossary

#

---

Specify immediate addressing mode for the next op-code generated.

BEGIN, UNTIL,

(Because of their close relationship, these words are covered in one glossary entry.)

They occur in a CODE definition in the form:

```
BEGIN, ... cc UNTIL,
```

At run time, BEGIN, simply marks the beginning of a sequence of code that is executed repeatedly. The corresponding UNTIL, assembles code that branches back to the "BEGIN, point" if the processor condition code register does not satisfy the condition specified by cc. On the other hand, when the condition code register does satisfy cc, no branch is taken at the "UNTIL, point" and execution proceeds to the following code.

CODE

A defining word used in the form: CODE <name> ... END-CODE

Creates a "smudged" dictionary header for <name>, sets the CONTEXT vocabulary to ASSEMBLER, and executes !CSP. See also END-CODE. CODE is in the FORTH vocabulary.

END-CODE

---

Exit the assembler by making the CONTEXT vocabulary equal to the CURRENT vocabulary. Perform error checking with ?CSP. If successful, "unsmudge" the code word being defined.

MI

--- n

A condition specifier: Minus.

NEXT

--- addr

A constant. The address of the inner interpreter. CODE routines usually end by a jumping to NEXT (or PUSHBA).

NOT

cc --- cc

"Inverts" the condition code that precedes it. Thus EQ NOT IF,

will execute the code after IF, if the zero flag in the condition code register is not set.

PUSHBA                    --- addr

A constant. The address of a routine that pushes the contents of accumulators B and A on the stack and jumps to NEXT.

UNTIL,

See BEGIN,

,X                        ---

Specify indexed addressing mode for the next op-code generated.

ORNL/TM-10657  
Dist. Category UC-506  
Instruments

## INTERNAL DISTRIBUTION

- |                    |                                                              |
|--------------------|--------------------------------------------------------------|
| 1. D. W. Bouldin   | 11. R. S. Wiltshire                                          |
| 2. H. R. Brashear  | 12-13. Central Research Library                              |
| 3. B. G. Eads      | 14. Y-12 Document Reference<br>Library                       |
| 4. D. N. Fry       | 15-16. Laboratory Records                                    |
| 5. J. M. Googe     | 17. Laboratory Records-RC                                    |
| 6. W. R. Hamel     | 18. ORNL Patent Section                                      |
| 7. D. W. McDonald  | 19. I&C Publications and<br>Information Processing<br>Center |
| 8. D. R. Miller    |                                                              |
| 9. C. A. Mossman   |                                                              |
| 10. R. W. Rochelle |                                                              |

## EXTERNAL DISTRIBUTION

20. Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations, Oak Ridge, TN 37831.
- 21-124. Given distribution under Category UC-506, instruments.

