This is to certify that the

thesis entitled

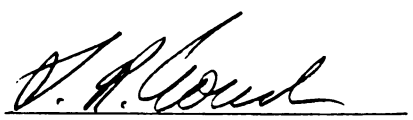
Investigation and Development of Electrothermal Atomization for Atomic Absorption Spectroscopy

presented by

Eugene Harlan Pals

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Chemistry



Major professor

Date May 26, 1978

INVESTIGATION AND DEVELOPMENT OF ELECTROTHERMAL ATOMIZATION FOR ATOMIC ABSORPTION SPECTROSCOPY

By

Eugene Harlan Pals

A DISSERTATION

Submitted to

Michigan State University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

Department of Chemistry

1978

ABSTRACT

INVESTIGATION AND DEVELOPMENT OF ELECTROTHERMAL ATOMIZATION FOR ATOMIC ABSORPTION SPECTROSCOPY

By

Eugene Harlan Pals

A computer-controlled electrothermal atomizer atomic absorption (EAAA) instrument has been developed which utilizes sophisticated software and hardware to automate all of the individual machine functions involved in a typical EAAA analysis. The computerized system contains a state of the art atomizer heating regulation device, an automatic sample dispenser, and a computer-controlled atomization cell positioning device.

The atomizer heating regulation device allows the operator to select any of four types of feedback regulation of atomizer heating: current, voltage, power, or radiation. The variable volume automatic sample dispenser is capable of rapidly and precisely delivering 1.0 to 5.0 μℓ sample solution to the atomizer surface, and this allows total EAAA measurement precisions of 1 to 3%. The automatic vapor cell positioner allows the precise displacement in two dimensions of the atomizer

relative to the optical axis of the instrument over a one inch range in each dimension for spatial studies of the atomic vapor plume.

The system software also provides for automatic execution of the simplex algorithm to optimize a chosen instrument response such as the magnitude of the absorbance signal. Any 2 to 4 of the 13 experimental parameters under computer control may be varied by this algorithm to achieve the optimum desired response. The computer-controlled EAAA instrument with its associated software has been shown to be a powerful and versatile tool for studies of fundamental EAAA processes. The simplex optimization technique is also shown to be a valuable tool in optimizing the performance of the EAAA instrument.

ACKNOWLEDGMENTS

During the time I was in graduate school many people have been kind to me when they could have been indifferent, have helped me when they had more important work of their own to do, and have somehow made the trials and tribulations of research not only bearable, but at times even enjoyable. There is no possible way to thank everyone who deserves credit, but a few people do deserve special mention.

First, my research preceptor, Professor Stanley R. Crouch, who managed to guide my research in the proper direction without stiffling me when I wanted to try something outside the master plan. I wish to thank Dr. Chris Enke for serving as my second reader, Dr. Tom Atkinson for rendering invaluable assistance in the production of many of the figures presented in this work, and also Dr. Andrew Timnick because he was such a pleasure to teach for and a good friend as well.

Also I must thank Dr. Eric Johnson for helpful discussions in the areas of electronics and computer programming and Marty Joseph for the use of his simplex routine and his help in applying it to the EAAA instrument. Jim Gano helped with some of the final stages of this work and also deserves acknowledgment. Special thanks go to my coworker, Dr. Dave Baxter, who is one of the

most gifted researchers I have ever met and without whom much of this work would have been impossible.

I would also like to express my gratitude to Coach Ron Schippers who taught me many things both on and off the football field and showed an interest in my academic career before, during and after my playing days at Central College.

To Gary, Leslie, Marty, Betty, Dan, Mel, Roy, Sandy, Jim, Clay, Rytis, Kathy, and all the other people at MSU that I have grown to know and love, thank you, my life would have been something less had I not had the privilege of knowing you.

Finally, and most important, I want to thank my mother, my father, my sister Eunice, and especially my sister Lois. Without them this work might have been possible, but such a success, unshared with them and unappreciated by them, would certainly have been an empty victory.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Since all fields of science are interrelated, developments in one area often profoundly effect other areas of science. Nowhere is this more clearly demonstrated than in the relationship of electronics to analytical chemistry. The electronic developments of the vacuum tube, photomultiplier tube, transistors, and integrated circuits have, in each case, permitted the manufacture of a new generation of analytical instrumentation and opened up new research areas. Most recently, large scale integration and microprocessor developments have been revolutionizing analytical instrumentation by integrating into the instrument computational, sequencing, and decision making tasks previously relegated to the operator.

Although the impact of computerization and automation on analytical chemistry has been profound, the present developments are just the tip of the iceberg. Already many research laboratories are equipped with computers which are much more powerful than any computer in existence 30 years ago. If the present pace of micro-minaturization of electronic circuitry continues it is not only conceivable, but indeed probable, that the research scientist in the not too distant future will have at his

disposal a laboratory computer as powerful as any of the largest computers now operating. The potential uses of such a computer system are mind-boggeling. And, the possibility that systems will be designed that are so gargantuan, intricate, and complex that it will be impossible for anyone but an expert to use them is very great. How could this new computer power be best used? Or, to rephrase the question in a more leading form - what should be the objectives of present research in this area?

One way to answer that question is to consider the past and present roles of the computer in the context of the classical scientific method (observation, hypothesis, experiment, theory or new hypothesis) and then extrapolate along the present path to the most reasonable and logical conclusion. Computers were first used in the experimental step to calculate results from raw data. Later, on-line computers were used in the actual execution and data gathering, and now it is quite common for a computer to control both the data acquisition and data analysis.

Computer curve-fitting programs are allowing the computer to enter the hypothesis and theory steps in a limited manner, but the most promising software developments in this respect are hueristic, "artifical intelligence" software and closed loop optimization software,

which can play a very active role in the theory and new hypothesis steps. Unfortunately, the recent trend in computer usage has been directed more toward the widespread application of existing techniques to previously non-automated experiments, rather than toward developing software which moves the on-line computer into areas of research other than just the experimental step. Because of the increased capabilities of the new generation of computers for routine on-line instrument interfacing, it is now possible to consider automating the experimental research process rather than just automating an instrument. This distinction bears further elucidation. Instrument automation entails development of the hardware and software necessary to allow an instrument to gather data without any operator intervention except to start the experiment. Automation of research involves the use of the computer not only to assist in the execution of an experiment and the analysis of the results, but also to assist in the process of either establishing a theory from the results or generating a new and reasonable hypothesis to test. Total automation of research may never be possible since it is difficult to conceive how a computer could be programed to have the intuition and serendipity of a good researcher, but the computer could be a very valuable aid in many areas where it is not now being used.

Electrothermal atomization has become a very popular technique in environmental and clinical laboratories in particular because of the very small sample sizes required and the extreme sensitivity of the technique for many metallic elements important in these fields. Although many electrothermal atomizer atomic absorption instruments have been interfaced to a dedicated computer or microcomputer, few have used the computer to its fullest capabilities. Many have not automated the sample introduction step so that the operator must constantly attend the instrument even during routine analysis.

The purpose of this work was to design and construct a highly automated electrothermal atomic absorption instrument that takes full advantage of the power of the system's computer. The automation of this technique involved the construction of several pieces of special purpose equipment. A positioning device was constructed to translate in two dimensions (with respect to the optical axis of the instrument) an atomic vapor cell containing the atomizer. A computer-controlled automatic sample dispenser, capable of accurately and reproducibly depositing microliter samples onto the atomizer, was constructed and has been extensively characterized in this work. The computer software necessary to drive the instrument is described. The computer-controlled instrument was also programmed to execute the simplex algorithm,

which is a sophisticated, time-efficient optimization technique. The results of some of these optimizations as well as the characteristics of the algorithm as applied to electrothermal atomization atomic spectroscopy are also discussed. The major significance of this work is that it demonstrates that once an instrument has been automated, an additional effort to expand the role of the system past the normal data acquisition and calculation assignments into sophisticated self-optimization tasks greatly enhances the utility of the instrument not only as a research tool, but also as a tool for practical analyses.

CHAPTER 1

HISTORICAL

A. ELECTROTHERMAL ATOMIZATION ATOMIC ABSORPTION SPECTROSCOPY

Atomic Absorption (AA) spectroscopy was introduced as an analytical tool by Walsh (1,2) in 1953 and Alkemade and Milatz (3,4) in 1955. The absorption phenomena had been observed in 1802 by Walston (5), but the only application of this technique outside astronomy before Walsh was the determination of mecury vapor in air (6). The near coincidental development by independent researchers of AA spectroscopy as an analytical measurement technique a century and a half after the discovery of the phenomena was due primarily to the development of good photoelectric detectors and the development of hollow cathode discharge tubes at about this time.

Most early work in AA spectroscpy utilized a flame as the atomic vapor cell. In 1959, L'vov (7) developed a resistively heated graphite furnace which could be used as an atomic reservoir. This work attracted little attention until the late sixties when Woodriff (8-14) and Massmann (15,16) introduced their own versions of graphite

furnaces. The technique rapidly became popular after 1970 when Perkin-Elmer Corporation produced the first commercial AA instrument equipped with a graphite furnace atomizer (17-19).

Since that time, many other companies have followed suit and introduced their own versions of EAAA instruments. One very significant variation on the design of electrothermal atomizers was the introduction of filament atomizers by West (20-23). Many types of base materials and physical structures have been used as filament atomizers. These include the graphite braid atomizer, which was introduced by Crouch and Montaser (24-27), and characterized by Baxter (28). This atomizer was used exclusively in this work. The application of EAAA and other non-flame techniques such as chemical vaporization and sample boat techniques to real analyses has been exhaustively reviewed elsewhere (28-30) and will not be reiterated here.

B. THE SIMPLEX METHOD OF OPTIMIZATION

In analytical chemistry, as well as daily life, one is constantly confronted with the task of optimizing the performance of some system, process, or device. Normally, this optimization is attempted by iteratively optimizing the response of each individual variable. The failure of this single factor at a time optimization

process was first pointed out by Box (31). Box and Wilson (32) presented the first organized discussion of response surfaces and hill-climbing theory in 1955. Later Box introduced the EVolutionary OPerations (EVPO) technique as a method of finding the response surface optimum. The occasional failure of this approach to find the true optimum was pointed out by Lowe (34).

In 1962 Spendley, Hext, and Hinsworth (35) introduced the simplex method of optimization. The simplex is a geometric figure of n+1 dimensions, where n is the number of factors (variables) to be optimized. This figure is moved across the response surface toward the optimum by the simplex paradigm. The response is first evaluated at each of the vertices and the worst (lowest response) vertex is reflected through the centroid of the opposite face of the simplex. The response at this new vertex is then evaluated, the new worst vertex determined and reflected to create a new simplex, and so on.

This paradigm has been modified a number of times, most notably by Nelder and Mead (36) and Denton (37). The Nelder and Mead paradigm is called the modified simplex. It allows the worst vertex to be displaced to any of four points on the line defined by the worst vertex. This allows the simplex to be expanded or contracted so that it can approach and converge on the response surface optimum rapidly. A more detailed description of the

modified simplex and optimization terminology is given in Chapter 5.

The Denton modification was named the super modified simplex. This modification allows the new simplex vertex to be located anywhere along the line defined by the worst vertex and the centroid of the opposite face of the simplex. The displacement distance is determined by an evaluation of the shape of the response surface in the region of the simplex. Three points are used in this evaluation, the response at the old worst vertex, at the centroid of the opposite face, and at the point obtained by reflecting the worst vertex through the centroid of the opposite face. The concave or convex nature of the response surface, as perceived by this three point evaluation, indicates whether the next move should be an expansion or a contraction. The distance of the move is determined by the overall slope of the response surface again as determined from these three points. This approach theoretically allows the simplex to move more rapidly than the modified simplex through plateau regions and also to converge on the optimum without seriously overshooting it.

The modified simplex algorithm has been used in chemistry for regression analysis in enzyme competitive inhibition (38), analysis of multicomponent spectra (39), dead-time estimation in chromatography (40),

chromatographic absorption isotherms (41), and pattern recognition (42,43). The considerations involved in the application of the simplex algorithm to numerical analysis are somewhat different from those involved when the technique is applied to the optimization of dynamic systems. Thus, the numerical uses of the algorithm need not be discussed in detail here.

Lowe (34), Long (44), and Deming (45-47) have all discussed the difficulties encountered when the simplex optimization technique is applied to dynamic chemical systems. Long (44) gives a particularly good discussion on how to choose the factors to be optimized, how to select the quantity to be optimized, and how to select the size and location of the starting simplex. Long also points out that noise on the response surface can cause moves of the simplex away from the true optimum, and that the simplex paradigm should be self-correcting for these false moves unless the noise is excessive and causes premature contraction of the simplex.

King (48) castigated some of the early users of the simplex optimization technique for misuses. One common error is the attempt to optimize coupled variables such as reagent concentration and reagent volume. These factors contribute multiplicatively to the more fundamental factor, which is the amount of reagent in the final solution. When two coupled variables are chosen

for optimization, the response surface contains a stationary ridge which defines the optimum combinations of these factors. Two other common errors are the incorrect definition of the response and the poor choice of factors to be optimized. In the example of the optimization of a spectrophotometric determination, the response is sometimes claimed to be sensitivity when in actuality the response optimized is the absorbance. The two may be assumed related with some validity only if one of the factors chosen for the optimization is not in some way related to the sample amount. If sample amount is chosen as one of the factors then either the response should be defined as the absorbance, or if the response is defined as sensitivity then the absorbance must be normalized for the amount of sample present. Several researchers have made this type of error (49-51).

Some applications of the simplex technique to chemistry are listed in Table 1. Several of these are of particular significance to this work and warrant more detailed discussion. Rippetoe, Johnson, and Vickers (57) performed the optimization of the signal-to-noise ratio of calcium emission with respect to arc current, slit width, slit height, and arc optical path on a DC plasma arc instrument. This appears to be the only direct application of the simplex technique to optimizing the precision of a measurement. Unfortunately no particulars

Table 1. Simplex Optimizations of Chemical Processes and Instrumentation.

System	Response Optimized	Ref.
p-rosanaline detn. of SO2	absorbance	44
AA detn. of Ca	sensitivity	52
continuous flow spectrometry	precision, minimum carry over interaction, enzyme activity	53
organic synthesis	reaction yield	54
Lieberman-Buchard detn. of cholestrol	sensitivity product stability	47
drug design	biological effectiveness	55
drug design	biological effectiveness	56
DC plasma arc Ca emission	signal-to-noise ratio	57
NMR spectrometer	field homogeneity	58
pulsed hollow cathodes	peak intensity	59
continuous flow spectrometry	sensitivity	60
anodic stripping voltametry	minimum analysis time	61
AE detn. of Ca	signal intensity	62

are given by these authors concerning the actual method of the determination of the signal-to-noise ratio or the behavior of the simplex paradigm in this type of optimization. Johnson, Mann, and Vickers (59) later reported the optimization of the peak intensity of a pulsed hollow cathode lamp, since the direct optimization of the signal-to-noise ratio was impossible due to the imprecision involved in the determination of the noise level (63). Parker, Morgan, and Deming (52) optimized the absorbance of a calcium solution with respect to the fuel flow rate, lamp current, burner height, and the volume of water in a graduated cylinder located at the end of the lab bench some distance from the spectrometer. The simplex converged on values for the water volume and lamp current even though they were later shown to have no effect on the magnitude of calcium absorbance. These workers suggest that the significance of factors should be tested after the simplex optimization by initiating a multiparameter factorial design study in the region of the optimum. Note that these workers optimized the calcium absorbance for a fixed amount of analyte and assumed this was related to the sensitivity and probably the precision of the measurement. Other statistical search methods have also been used to optimize the absorbance response in AA analysis (64-67) with results not dissimilar to those obtained in the simplex optimization.

One study listed in Table 1, that is of particular interest because of its unique approach, is the optimization of analysis time in anodic stripping voltammetry by Thomas, Kryger, and Perone. The controlled factors for this study were the plating time and scan rate. The optimization was carried out within the limits imposed by a "performance criterion" which was a user-defined minimum acceptable value for either the signal-to-noise ratio or the signal-to-background ratio. Also, the computer was used to calculate some preliminary constants from theoretical equations. These constants were necessary for the evaluation of whether the simplex might be moving into a region where the selected "performance criteria" would not be satisfied. Unfortunately, although the work is exemplary in a number of areas, the authors chose to use their own terminology rather than the standardized vernacular of response surfaces and optimization techniques. The response optimized is actually a combination of the analysis time and the chosen "performance criteria". The correct term for these "performance criteria" is constraint or boundary, since a violation of the constraint is signalled to the simplex algorithm as a penalty to the delay time response. This work also involves the unusual feature in that one of the parameters to be optimized has only 11 discrete levels. The computer was used to round off the simplex suggested value of this

parameter to the nearest of the discrete levels, and the authors maintain that this caused no significant problems if the initial simplex was large in this dimension.

In summary, the simplex technique has been widely used and misused. Most optimizations have only contained two, three, or occasionally four significant factors so that the number of iterations required were within reason. The response chosen to be optimized is often some response, such as absorbance, which is then assumed to be related to the response that would really be the most desirable to optimize, such as the measurement sensitivity or precision.

CHAPTER 2

INSTRUMENTATION

A. GENERAL DESCRIPTION

A block diagram of the computer controlled Electrothermal Atomization Atomic Absorption (EAAA) instrument is given in Figure 1. The computer is a PDP 8/e equipped with 16k of mainframe memory and an extended arithmetic element. The computer peripherals used in this work include: a Sykes 7000 dual floppy disc, a DEC RK05 cartridge disc, a DEC LA-30 Decwriter used as a line printer, and an ADDS Consul 980 CRT used as the system terminal.

Non-computer commercial equipment included: a GCA McPherson EU-703-70 hollow cathode discharge tube (HCDT) power supply, assorted Jarrel-Ash and Westinghouse hollow cathode discharge tubes, a GCA McPherson EU-700/E monochromator, a GCA McPherson model EU-701-30 photomultiplier module with an RCA 1P28A or Hamamatsu R166 solar blind photomultiplier (PMT), a model 427 Keithley current amplifier, and a Tektronix model 5043 oscilloscope.

The remainder of the instrument is composed of modules built by my coworker, Dr. Dave Baxter, and myself. The modules discussed in this chapter include the atomization

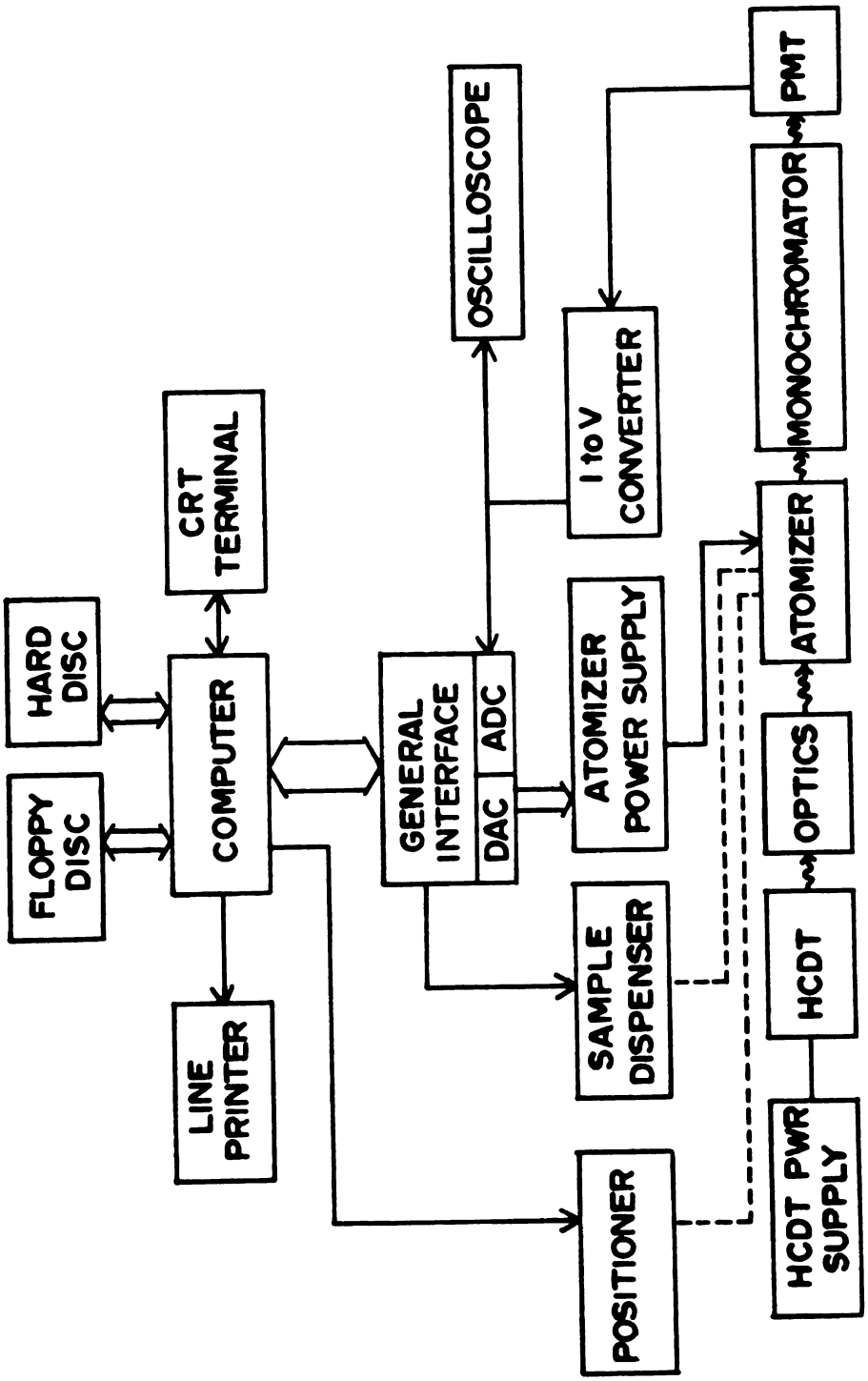


FIGURE 1: BLOCK DIAGRAM OF THE COMPUTER CONTROLLED EAAA INSTRUMENT.

vapor cell, the atomizer power controller, the atomizer vapor cell positioner, the general interface, and the instrument optics. The automatic sample dispenser is discussed in Chapter 4.

B. ATOMIZATION VAPOR CELL

A simplified scale drawing of the atomizer vapor cell is shown in Figure 2. The graphite braid atomizer (Union Carbide Corp., Parma, OH) is held in the jaws of a brass holder at each end. When a screw in the upper jaw of the holder is loosened, the jaw may be slid away from the atomizer so the atomizer can be removed and replaced. The power connections are made to the bottom of the lower jaw near the outer end of the holder. It was necessary to mount the atomizer holder on brass heat sinks since prolonged heating of the atomizer would heat both the holder and the material it was mounted on enough to deform plexiglass. The brass heat sinks are electrically isolated from the steel support structure by a layer of plexiglass. The channel formed by the brass heat sinks on two sides, and a plexiglass plate on the other two sides, contains the sheath gas flow apparatus. This apparatus was fashioned from close packed glass capillaries glued together and was supported on a ridge along the inside walls of the plexiglass box. The laminarity of gas flow (28) from this type of apparatus

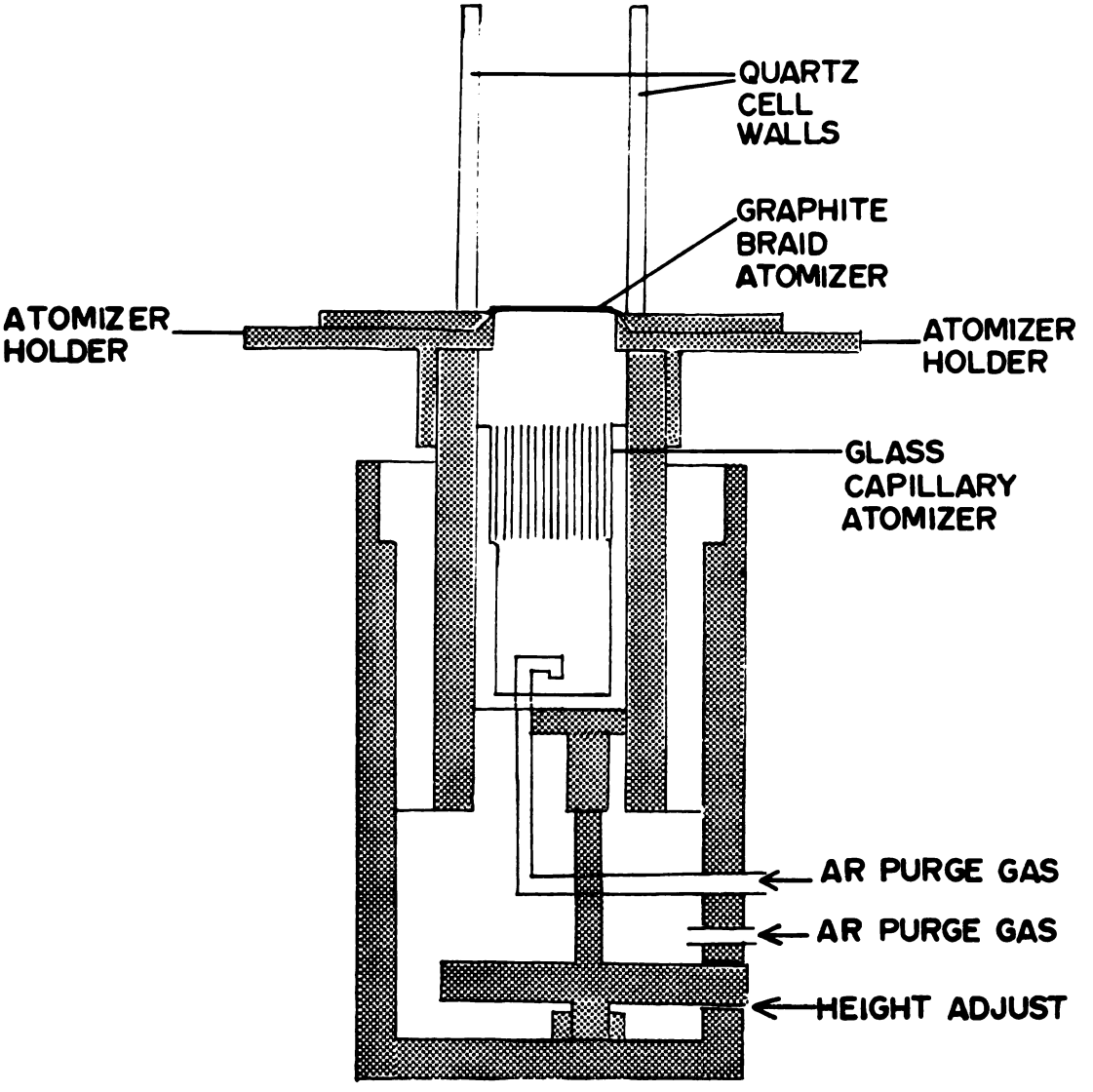


FIGURE 2: THE ATOMIZER VAPOR CELL

far exceeds that of any previous designs which had holes drilled in a metal block. The position of this sheath gas flow apparatus relative to the braid could be changed by rotating a thumbwheel mounted at the lower end of the support structure. The axle on which this thumbwheel is mounted and the base plate of the sheath gas apparatus are threaded so that rotation causes movement of the sheath gas apparatus relative to the atomizer. To change the orientation of the atomizer relative to the optical axis of the instrument, the braid holders and associated parts may be lifted from the support structure, the sheath gas apparatus may then be removed by rotating the elevation thumbwheel until the sheath gas apparatus is unfastened. The brass base plate of the sheath gas apparatus is then removed and mounted in an alternate position. The whole system may then be reassembled with the atomizer orientation 90 degrees different from the original position.

Along with the novel glass capillary gas flow shaping apparatus, a unique feature of this design is the capability to configure the system so that the atomizer may be mounted parallel to the optical axis with the collimated light beam directed at grazing incidence just above it. The advantages of parallel mounting include a reduction in the amount of atomizer continuum background radiation viewed by the monochromator, a reduction in the critical nature of the location of the sample on the

atomizer, and a sensitivity enhancement due to the increase in the absorption path length. The effects of sample size should also be reduced when the atomizer is mounted parallel to the optical axis, since sample spreading will not cause any sample to be vaporized from a region of the atomizer that is not under observation. To achieve this parallel configuration it was necessary to design the vapor cell walls from optical flat quartz plates (ESCO Optics, Oak Ridge, NJ) and to design the braid holders so that no part of them would occlude any source radiation. An additional restriction on the design of the holders was that as little as possible of their structure should protrude from the walls of the vapor cell into the gas flow. This is critical in maintaining the good laminar gas flow generated by the glass capillary array.

A phototransistor (now shown in Figure 2), which is used to measure the blackbody radiation of the atomizer, is mounted on top of one of the plexiglass side walls at about a 45 degree angle to the atomizer, and views the atomizer through one of the quartz walls of the vapor cell.

C. OPTICS

Atomic vapor mapping experiments with the EAAA instrument require that the source optics yield a high intensity, collimated, small diameter beam of radiant energy, whereas

the optimal optics for other experiments, such as detection limit studies, may be somewhat different. For this reason the optics were designed so that they could be arranged in several different configurations. The base of the instrument is a GCA McPherson EU 703 optical rail, and all components are fastened to it in some way. A diagram of the EAAA instrument (simplified by omitting the positioner) is shown in Figure 14, Chapter 4.

Three different lenses can be used. Two are 2" diameter, 2" focal length quartz lenses with precision lap finishes, the third is a 1/2" diameter, 1/2" focal length quartz lens. Each lens is mounted in a Teflon ring and is fastened to a 1/2" aluminum shaft by a gimbal mount so that height and tilt adjustments may be easily made. One fixed position receptacle for the lens mount is attached to the atomization cell positioner. Also on this positioner is a receptacle mounted in a channelled track on the positioner, so that this lens may be moved over a 3" range along the optical axis. The third receptacle is a free standing mount and may be placed anywhere along the optical axis. The atomization cell positioner itself is designed so that it can be easily moved to any position along the optical axis.

The two most useful configurations are shown in Figure 3. Configuration A has a better collection efficiency and light throughput than configuration B and is thus

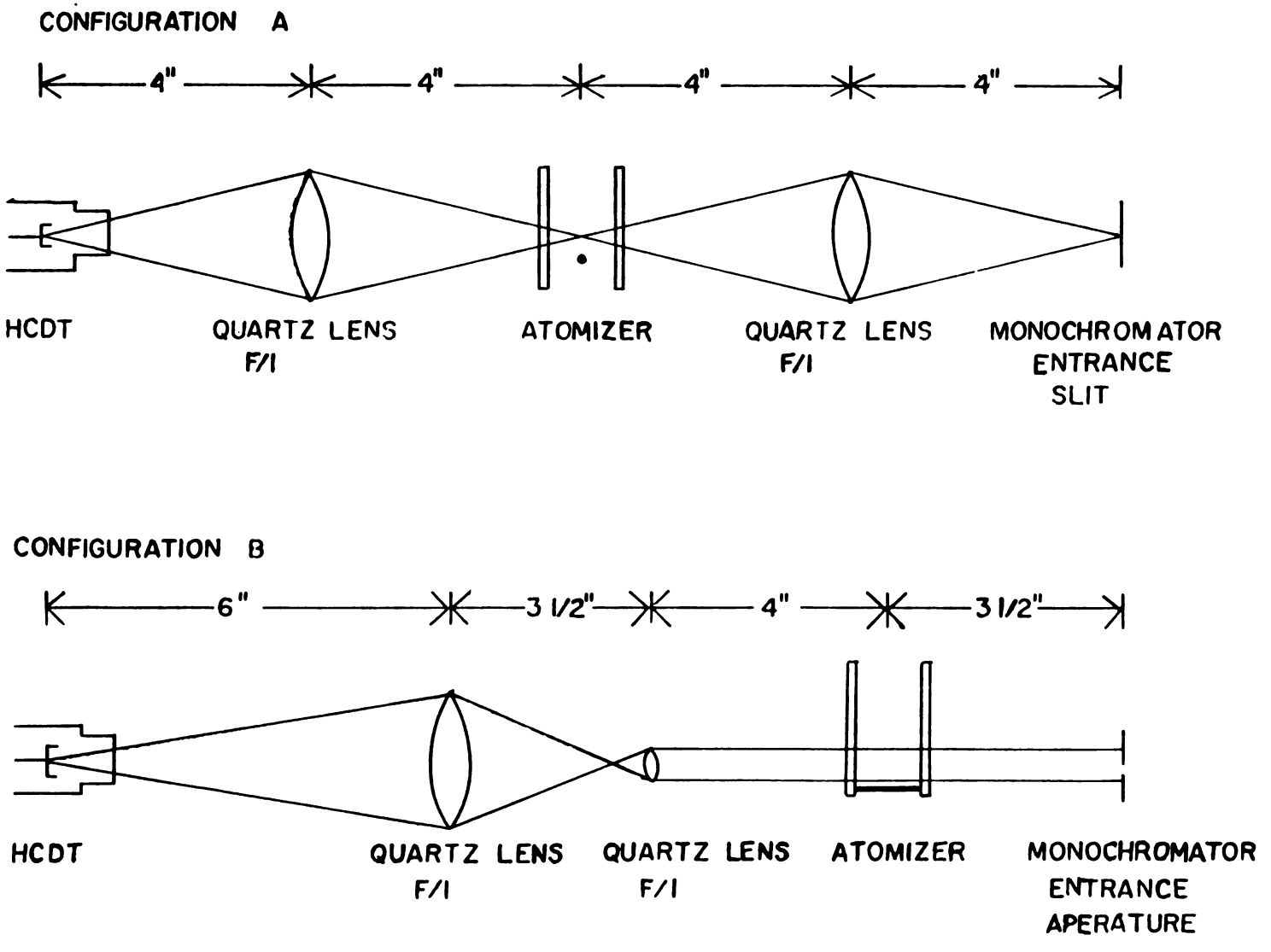


FIGURE 3: OPTICAL CONFIGURATIONS FOR THE EAAA INSTRUMENT.

preferred for detection limit studies. Also, since the observed area above the atomizer is smaller than configuration B, this arrangement is useful when volatilizing analytes that have a short half-life as free atoms in the atomic vapor. Note, however, that this configuration precludes mounting the atomizer parallel to the optical axis of the instrument so that the advantages of parallel mounting described in the previous section may not be realized.

When mapping experiments are required, the optics must be set up in configuration B, but the atomizer may be mounted either parallel to or perpendicular to the optical axis. Also note that although this configuration produces a beam several mm in diameter and with a small divergence, the area of the vapor cell observed is actually defined in the horizontal plane by the monochromator slit width and in the vertical plane by the diameter of the aperature (typically 3 mm) at the entrance slit of the monochromator.

D. ATOMIZER POWER CONTROLLER

1. Introduction and General Overview

One of the critical parameters in EAAA is the temperature of the atomizer during the desolvation, ashing, and atomization heating steps. In the atomization heating

step, the rate of temperature increase, as well as the final temperature, can have a profound effect on the size and shape of the absorbance signal. The purpose of the atomizer power controller is to provide a good, stable temperature related regulation of the atomizer power supply and to allow easy comparison of this regulation method to the more conventional regulation methods.

The temperature of the atomizer is related to the power input, power loses, and the mass of the atomizer. Power input is easily defined as the product of the current through the atomizer and the voltage drop across the atomizer (27,29). As the atomizer ages its mass and resistance change due to sublimation of the graphite during heating. If a conventional power supply is used to regulate either the current through or the voltage drop across the atomizer, the final temperature for a requested current or voltage will change as the atomizer ages (27, 29). Regulating the power dissipated (power regulation) during a heating step should offset the effect of changes in resistance of the atomizer due to aging, but will not counteract the effect of atomizer mass changes. It would be even more desirable to choose a feedback signal that was directly related to the atomizer temperature. When moderate to high heating levels are desired this type of regulation is possible (27,29) by monitoring the blackbody emission of the graphite atomizer (radiation

regulation). The circuitry to handle these functions will be described in this section along with the circuitry which provides the hydrogen diffusion flame. Characterization of the different methods of regulation as they relate to overall EAAA measurements were carried out by Dr. David Baxter and are described elsewhere (28).

A simplified diagram of the power controller and its relationship to the general interface and the atomizer power supply is shown in Figure 4. The raw power supply and water cooled power transistor bank were built by the MSU Chemistry Department electronics shop for Dr. Akbar Montaser (27,29). Regulation is achieved through a standard Darlington transistor configuration and a control OA. One unfortunate characteristic of this design is the amount of power wasted in the ballast and sensing resistors. Since the atomizer resistance is about one ohm, and the sum of the resistances of the ballast and sensing resistors is about .8 ohms it is obvious that much of the power supplied by the raw power supply is dissipated in these resistors and is not available for heating the atomizer.

All of the feedback signals necessary for current, voltage, power, or radiation regulation are brought into the controller box. In the first section of circuitry they are modified by amplification, level shifting, etc., and then the proper signal is selected to be passed to

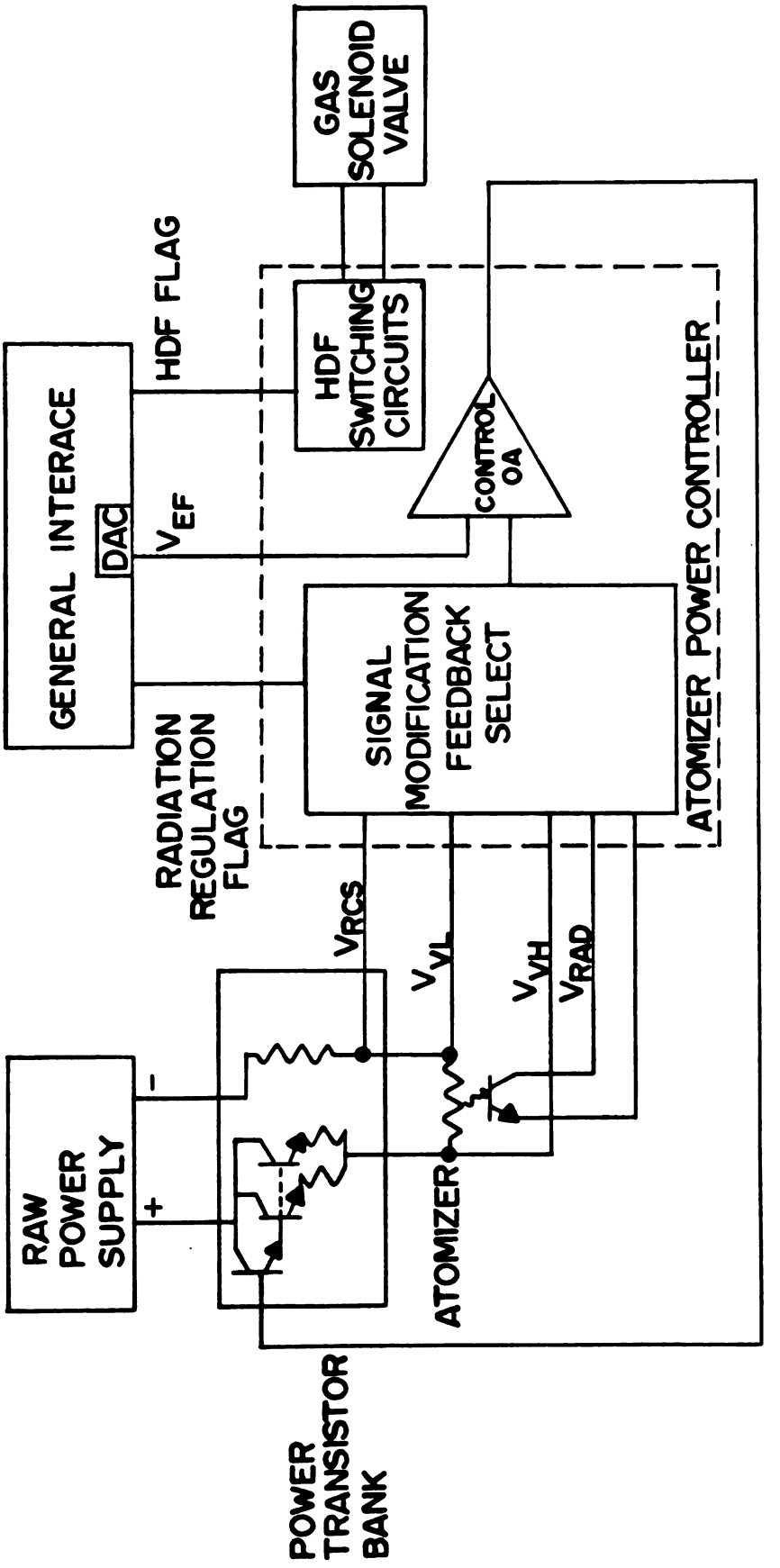


FIGURE 4: OVERVIEW OF ATOMIZER POWER REGULATION CIRCUITRY

the control operational amplifier where it is compared with the reference voltage. This OA then sends an appropriate signal to the power transistor bank to correct the current through the atomizer.

Signals from the general interface circuitry provide the heating level reference voltage, the signal to switch to radiation regulation, and the signal to switch on the gas for the hydrogen diffusion flame (HDF). Front panel switches include the following: power on-off, kill atomizer power, hydrogen diffusion flame override switch, and two regulation mode selector switches for the heating steps. The last two multi-position switches allow a different regulation mode to be selected for the ash and atomize, or just the atomize heating step or steps. Thus it is possible to select current, voltage, or power regulation for the desolvating step and any of these or radiation regulation for the ashing and atomizing, or just the atomizing step. A precision potentiometer with a locking mechanism allows the adjustment of the sensitivity of the radiation feedback circuitry relative to the other regulation methods and may be used to counteract differences in phototransistor sensitivity if the phototransistor has to be replaced or mounted in a new position. Back panel banana jack outlets provide monitors of the current, voltage, power, and radiation levels regardless of which regulation method is actually being used. Also on the

back panel are the ports for receiving the interface signals, the power outlets to the gas valve switch, and the connectors necessary to communicate with the power transistor bank, phototransistor, and the current sensing resistor.

2. The Control Operational Amplifier

The control OA section of the power controller circuitry is shown in Figure 5. The first transistor in the Darlington cascade is actually in the power controller box and not in the power transistor bank. The base of this transistor is connected to the output of the control OA (OA1) and regulates the current from the raw power supply into the base of the power transistors in the Darlington cascade. This OA operates on a current null comparison and its inputs are protected from damage by overvoltage spikes by two diodes. Also the resistor, capacitor circuit serves to the limit frequency response and thus power supply oscillation. OA2 is configured as a difference amplifier and as such provides the dual purpose of buffering the input reference voltage and providing some protection from ground loops. The grounds of the interface circuitry and the power controller circuitry are connected, but a small potential difference might exist and cause not only imprecision but also oscillation in the power supply. The 1 kΩ potentiometer allows some adjustment of

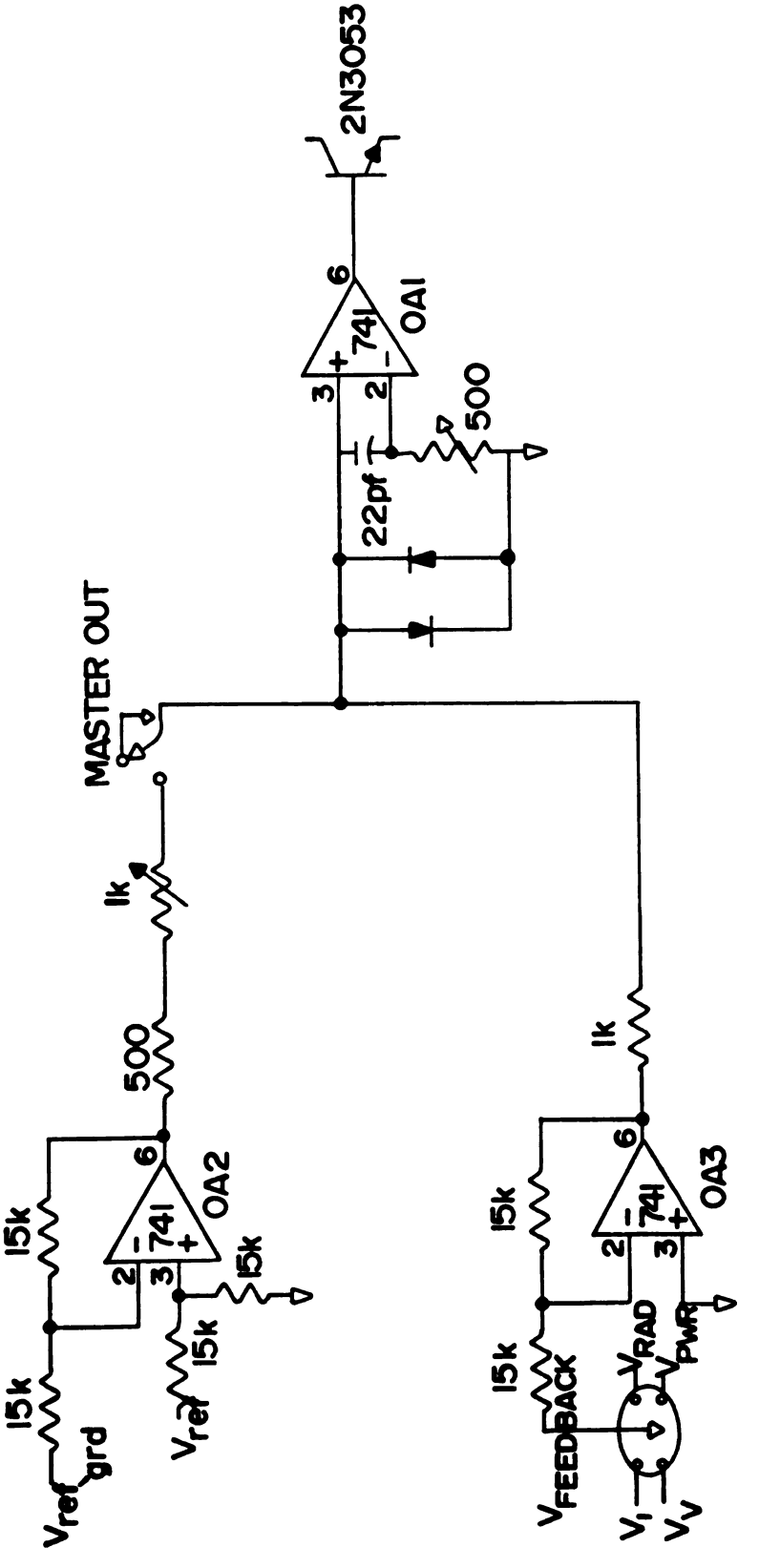


FIGURE 5: CONTROL OA CIRCUITRY.

the atomizer heating level relative to the voltage reference input. The feedback signal is fed from the front panel regulation mode selector switch to OA3 where the polarity is reversed but the magnitude is unchanged.

3. Current, Voltage, and Power Regulation Circuitry

The current regulation is quite straight forward and is shown in Figure 6 along with the voltage and power regulation circuitry. OA1 buffers the voltage signal from the sensing resistor that is located in the power transistor assembly bank. The output of OA1 is sent to the front panel mode select circuitry and the back panel current monitor output.

In previous designs the voltage drop across the atomizer was monitored with an instrumentation amplifier (27, 29). However, a difference amplifier (OA1) is quite sufficient since the input voltages are not large, and loading of the voltage source is not possible in this case. The potentiometer on the non-inverting input was adjusted so that the CMRR of the difference amplifier was maximized. Again the output voltage for the voltage regulation circuitry is sent to the front and back panels.

The power regulation circuitry utilizes an integrated circuit (IC) four-quadrant multiplier to compute the product of the current through and the voltage drop across the atomizer. The analog multiplier contains two on-chip

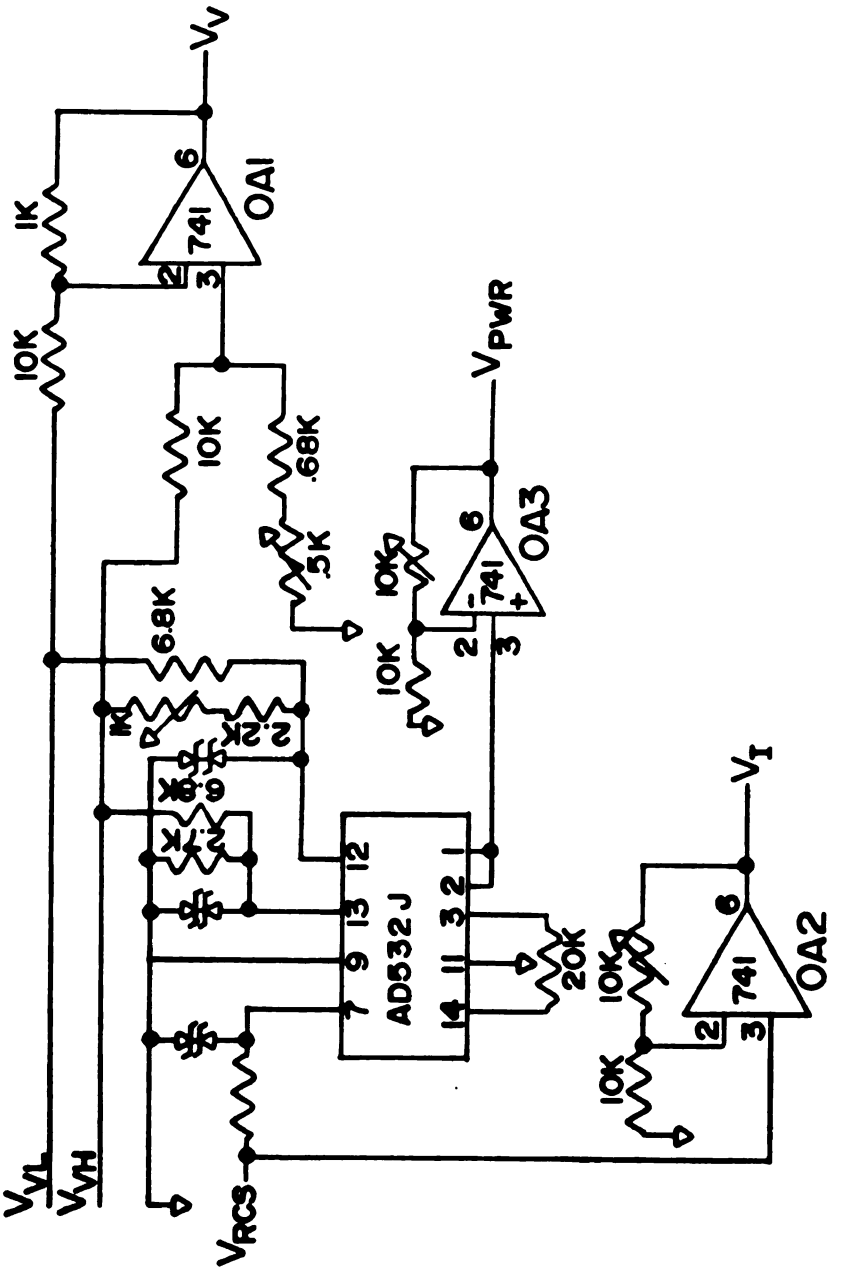


FIGURE 6: CURRENT, VOLTAGE, AND POWER REGULATION CIRCUITRY.

difference amplifiers. The multiplier's transfer function is the product of the differences of the voltages appearing at the difference amplifiers inputs divided by ten. The atomizer current is represented by the voltage drop across the sensing resistor, and this voltage is presented to one input of one of the difference amplifiers. The series resistor and the head-to-head Zener diodes protect the analog multiplier chip from the overvoltage damage. The inputs to the second difference amplifier are similarly protected and are used to monitor the voltage drop across the atomizer. At these inputs the two voltage divider networks scale the signal voltages to the appropriate levels and again a trimpot is provided to maximize the CMRR. The 20 kΩ potentiometer is used as a fine adjust on the output voltage level. The output of the analog multiplier is buffered by a gain-adjusting OA (OA2) before being sent to the front and back panels.

4. Radiation Regulation Circuitry

The radiation regulation circuitry is shown in Figure 7. The current output of the phototransistor, which is mounted to observe the atomizer blackbody emission, is converted to a proportional voltage by OA1, and the gain adjusted and the polarity reversed by OA2. Since this regulation mode is not applicable at low temperatures because the atomizer emission level is very low, a

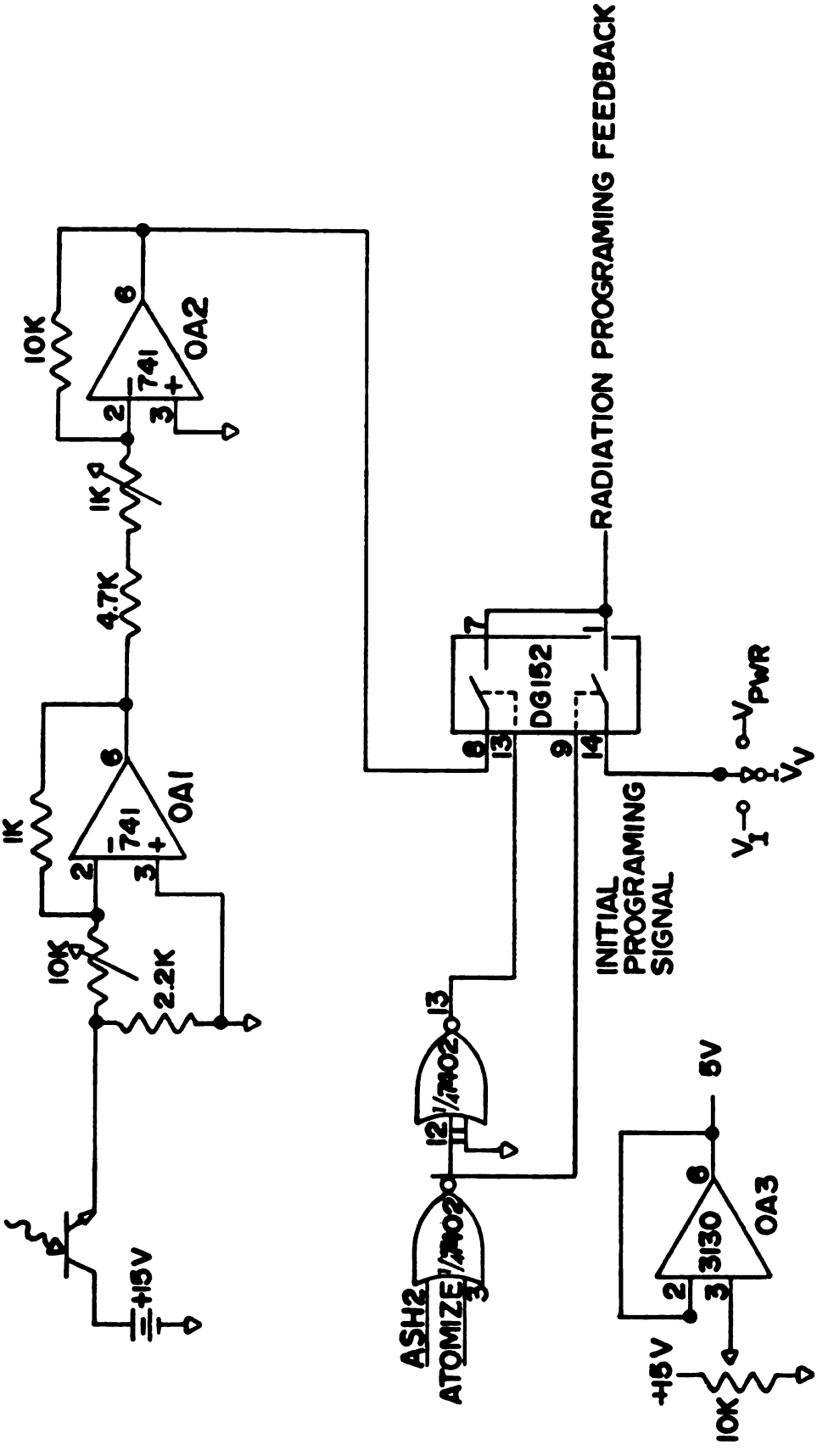


FIGURE 7: RADIATION REGULATION CIRCUITRY.

switching circuit is necessary to introduce another control mode for these temperatures. This switching circuit allows the desolvation, or desolvation and ashing, heating steps to be regulated in a different manner from the following heating step or steps. The switching is accomplished by an IC voltage switch controlled by a signal from the logic circuitry. One input of the voltage switch is from the regulation mode select for the initial heating steps and the other from the front panel regulation mode selector for the regulation mode for the final heating steps. The grandiose logic circuitry consists of one 7402 quad NOR state, which is used to provide a switch signal to each of the analog gates on the voltage switching IC. Since this is the only logic chip in the entire circuit, a cheap, but effective, DC to DC power supply was constructed from OA3. Again, the radiation feedback signal is fed to both the front panel mode select switch and a back panel monitor outlet.

5. Hydrogen Diffusion Flame Switching Circuit

The circuit necessary to switch the line voltage solenoid gas valve for the hydrogen diffusion flame is shown in Figure 8. The transformer, rectifier bridge, and filter capacitor form an unregulated 10 V DC power supply for the triac trigger circuit. When the hydrogen diffusion flame (HDF) is requested, the interface sends

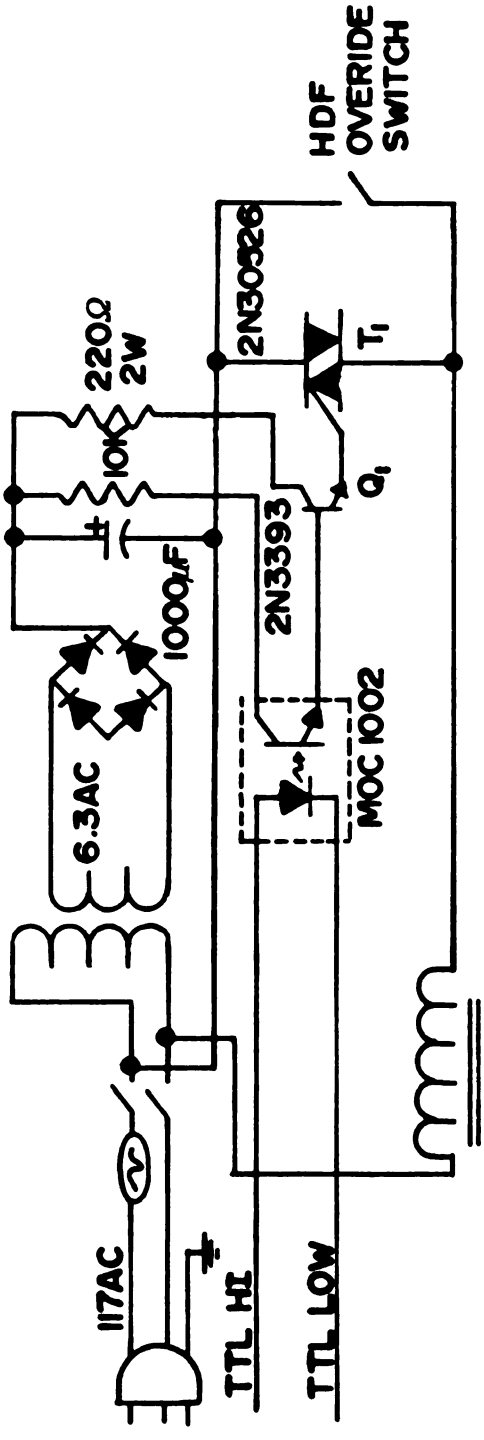


FIGURE 8: HYDROGEN DIFFUSION FLAME SWITCHING CIRCUIT.

a TTL signal to the optical coupler and causes the photodiode to emit. The phototransistor receiver is turned on when the light is received, and the current from it is boosted by transistor Q1 to a level that will turn on triac T1. The optical coupler is a safety device that serves to isolate the digital circuitry in the interface from the line voltage necessary to operate the solenoid gas valve. When the triac is triggered, the solenoid gas valve (Cat. No. 2-12, General Valve Corp., E. Hanover, N.J.) opens and allows the hydrogen gas to mix with the argon flowing into the atomization cell. The H_2 OVERRIDE switch is wired in parallel to the triac switching circuit and thus may be used to actuate the gas valve independent of the remote switching circuit. This is useful when the rotameter that controls the H_2 gas flow rate needs to be adjusted. The sudden switching of hydrogen gas causes a surge of gas through the atomization cell. This effect can be mediated somewhat if some glass wool is packed into the H_2 gas lines to provide some backpressure. The hydrogen reacts with the traces of oxygen in the argon sheath gas only when the atomizer reaches a dull red heating level (1000-1200°C). Thus it is necessary to exercise caution when the H_2 flame is used. If the flame consistently does not light, the laboratory may experience a sudden violent metamorphosis as this accumulated, unspent H_2 reacts rapidly to form water. Under normal conditions,

the amount of H_2 used is less than 1 ℓmin and the H_2 is switched on only during atomization. The diffusion flame produced occupies the entire height of the quartz walls of the atomization chamber. Fortunately, the flame is relatively cool and does not cause extensive heating of the cell walls. Also, it is nearly totally optically transparent. In fact, often the only real visual evidence that the flame has been ignited is the observation of the Schlerien effects it produces.

6. Prospectives

Much of this circuitry was constructed using archaic techniques and components and, because of this, suffers from contact resistance and other problems. Still, it has performed quite well. The major drawback of this design is that the useful range of the temperatures under radiation regulation is limited by the dynamic range of the phototransistor. This means that to have acceptable regulation at high temperatures, a neutral density filter must be inserted in front of the phototransistor to decrease the amount of light it observes. Ideally, radiation regulation would have at least two separate photosensitive elements; one for the low temperature range and another for the high temperature range. A phototransistor is perhaps the best for the low temperature range, since

it has good sensitivity. But, in the high temperature range a photodiode should be used because it has a faster response time and a wider dynamic range.

E. POSITIONER

1. Introduction and General Overview

The positioner is the device which displaces the atomization cell in two dimensions with respect to the optical axis of the instrument and thus allows the observation of the atomic concentration in any locale within the atomization cell. This device consists of two physically separate modules. One module contains the interface, general logic circuitry, stepper motor power supplies, and front panel controls. The second module (shown in Figure 9) consists of an aluminum superstructure on which is mounted the translational stepper motors, an x and a y translational stage, and the atomization cell with its associated optics. The positioner can operate in either a local mode or under computer control. Details concerning the operation of the logic circuitry can be found elsewhere (28,68) and will not be discussed here.

2. Physical Structure

Two translational stages (Cat. No. 420-51, Newport Research Corp., Fountain Valley, CA) and two stepper

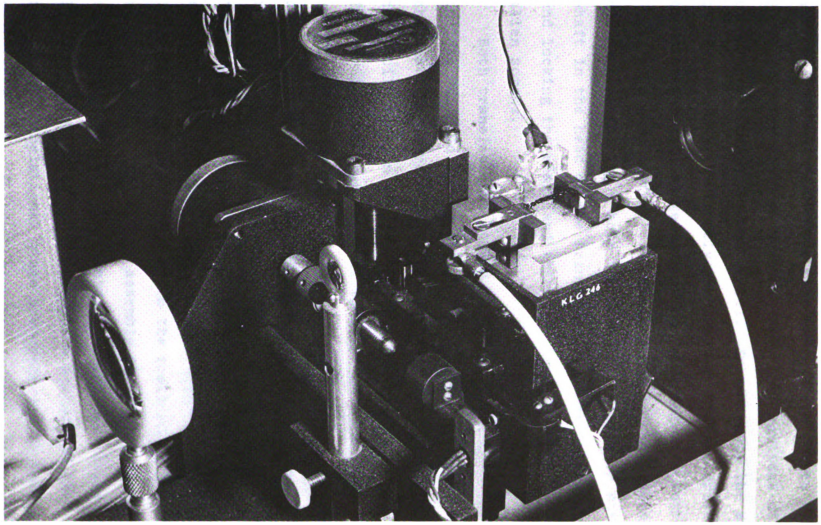


Figure 9. Illustration of the Positioner and the Atomization Vapor Cell.

motors (Cat. No. 23D-6102A, Computer Devices, Santa Fe Springs, CA) accomplish the horizontal and vertical displacement of the atomization cell. In each case the stepper motor shaft is directly coupled to the screw drive of the translational stage so that no imprecision is introduced by a gear system. The direct coupling was accomplished by drilling a hole the size of the stepper motor shaft in the head of the translational stage lead screw and locking the shaft and the screw together with two allen screws.

Both translational stages are perpendicular to the optical rail and are mounted side by side. The side of the base structure of the atomization cell is fastened to the mobile section of the vertical translational stage. The vertical stepper motor is fastened to the immobile section of the vertical translational stage and, as the stepper motor is pulsed, the lead screw which drives the translational stage rotates inside a nut attached to the mobile part of the translational stage. Thus, as the stepper motor is pulsed, the mobile section of the translational stage moves relative to the position of the immobile section. The entire assembly is mounted on the mobile section of the horizontal translational stage. The horizontal motor and the immobile section of the horizontal translational stage are connected to the positioner superstructure which is, in turn, fastened to

the optical rail. As in the vertical section, the horizontal stepper motor is secured to the immobile section of the horizontal translational stage. As the motor is pulsed, it turns the lead screw which causes the mobile section of the translational stage to move relative to the immobile section. Optical interrupters and occluders (mounted on the atomization cell, translational stages, and the positioner superstructure) are used to define the limits of travel of the translational stages and to provide a reliable reference position. The positioner superstructure is fastened to the optical rail in such a manner that it is possible to move it anywhere along the optical axis of the instrument. This structure also contains one fixed position and one mobile lens mount in-line with the entrance slit of the monochromator.

3. Remote and Local Operation

When the front panel mode switch is set to LOCAL, the other front panel switches are activated. The RESET switch causes the positioner to drive both the horizontal and vertical translational stages to the reference point defined by the optical interrupters described previously. This resets the counters in the logic circuitry to zero and, as further movements are made, these counters are used to point to the present position of the atomization

cell.

To move the atomization cell away from the reset position, numbers, which correspond to the desired horizontal and vertical destinations, are entered into the front panel thumbwheel switches, and the GO switch actuated. This causes the motors to begin concurrently to seek the desired location. If the direction of the movement is positive, the motors will move slightly past the desired location and then return to it. This means that the final direction of movement is always negative and, whatever slop there may be in the drive mechanism, is thus corrected.

The thumbwheel switches are 4 digits wide, but the logic circuitry accepts only 12 binary bits (a concession to the 12-bit computer used in the remote operation mode), so that legal decimal locations are from 0 to 4095 in both the horizontal and vertical directions. If a destination outside the legal limits is requested, a fault condition occurs, which lights the front panel FAULT LED and aborts the movement. To correct this fault condition, a reset must be executed. The position horizontal 0 and vertical 0 is the reset position, and the position horizontal 2048 and vertical 0 locates the atomizer just beneath the path of the absorbing light beam. This latter location is also used as the place for sample deposition when automatic sample deposition is desired. The actual physical

displacements that correspond to these user units can be easily calculated since each user unit corresponds to 2 stepper motor pulses, the stepper motors require 200 steps per revolution, and the lead screws are machined to have 40 threads to the inch. Thus, the end-to-end travel in both the horizontal and vertical directions is limited to 8192 pulses (1.024 in).

If the front panel mode switch is set to the LINE position, the other front panel switches are deactivated. Commands are then accepted only from the computer. The possible computer commands are GO, RESET, skip on flag, clear flag, enable interrupt, disable interrupt, load horizontal destination register, and load vertical destination register. The GO and RESET commands function exactly the same in the remote mode as they do in the local mode, and the load location register commands serve the same function as loading the front panel thumbwheel switches. Now, however, it is impossible to load an illegal destination since the locations are transferred in 12-bit binary. The skip-on-flag and clear flag commands are necessary so that a new command will not be accepted before the previous command has been completed. The positioner interface was constructed so that it could be interrupt driven. However, because of the timing and position constraints imposed on the positioner by the automatic sample dispenser and the data acquisition system, no practical

time savings would be realized by operating the device under interrupts. Thus, the added software complexity of an interrupt system could not be justified. A more detailed discussion of design concepts of the positioner is given in Appendix B and a very detailed description of the positioner's electronic circuitry is also available (28).

F. GENERAL INTERFACE

All communications to and from the computer with the EAAA instrument occur through the general interface, except for the positioner which contains its own interface. The interface functions may be conveniently divided into five categories: automatic sample dispenser, analog-to-digital conversion, digital-to-analog conversion, general purpose flags, and input switch register. An overview of the automatic sample dispenser interface is given in Chapter 4 and is discussed in great detail elsewhere (28).

Voltages representative of the photomultiplier photocurrents are quantized by an analog-to-digital converter (Cat. No. ADC-HY12BC, Datel Systems, Canton, Mass.). This 12-bit, hybrid, successive approximation converter has a conversion time of about 8 μs and a maximum non-linearity of ±1/2 LSB. In practice the quantization error due to the 12-bit resolution of the converter was observed only

in the measurement of dark current and was small enough so that it can be discounted as a significant source of error. The maximum conversion rate was about one order of magnitude faster than the software could handle the data. Even so, the software-limited conversion rate was still more than an order of magnitude greater than the slowest rate acceptable for an accurate integration of even the fastest absorbance signals. Although a converter of this speed is unnecessary for normal integration of absorbance transients, kinetic studies and studies of the rate of atomizer temperature increase might require a fast converter such as the one used.

The digital-to-analog converter (Cat. No. DAC-HY12BC, Datel Systems, Canton, Mass.) is used to supply the reference voltage for the atomizer power supply controller. This 12-bit hybrid converter has a conversion time of 3 μs and a maximum non-linearity of ±1/2 LSB. Again these specifications are much better than required by this particular application. For example, in radiation regulation, the resolution of this converter (2.5 mV) corresponds to a temperature resolution of 1.0° to 3.0°C (the actual resolution depends on the particular temperature region). Atomizer temperatures determined by optical pyrometry have a precision of at best ±25°C. Furthermore, because of the physical nature of the atomizer, the temperature in some regions on the atomizer may be more than 25°C different

from the nominal temperature. It is important that the atomization temperature be stable and reproducible, but for most elements the absolute value of this temperature may be any value within a several hundred degree range and still not affect the precision and accuracy of the measurement. The converter's slew rate of 20V/μs and settling time of 3 μs far exceed the response time of the atomizer heating circuitry and in no way limit the time it takes for the atomizer to reach its final temperature. The hardware associated with the three general purpose flags is composed of three flip-flops and associated circuitry. It is so routine that it does not warrant discussion here. The user input switch resister is an array of six single-pole, single-throw switches in a dual-in-line package on the front panel of the general interface box. On computer command, the status of these switches is read into the 6 most significant bits of the accumulator. Again the interface circuitry is of a common garden variety and does not warrant detailed discussion here, since it has already been discussed in detail by Dr. Baxter (28).

CHAPTER 3

PROGRAMMING THE COMPUTERIZED ELECTRO-THERMAL ATOMIZER ATOMIC ABSORPTION INSTRUMENT

A. INTRODUCTION

The electronic revolution of the 1960's and 1970's caused a significant increase in the availability of computers to chemists. At first chemists used computers only in the passive role of data reduction. Typically experimental data was manually transformed into a computer compatible form such as punched cards and physically transported to the site of a large central computer for data analysis. This large computer generally had a high level language such as FORTRAN available if the chemist had chosen to do his own programming, or sometimes he might even be able to use a routine supplied by the computer vendor. When the computer price structure changed so that it was reasonable to dedicate a computer to one instrument the scope of the chemist's software problem changed significantly. Now the computer could play an active role in the experiment including sequencing events and controlling data acquisition. More often than not computers in this price range had no laboratory oriented high level

language and even if one was available special control functions and data acquisition functions required routines which had to be written in assembly language. Now whenever a chemist found it necessary or desirable to modify an existing computerized instrument, computerize an existing instrument, or to build a totally new computerized device, he needed to deal with programming on a much more complex level. The problem was attacked in one of three ways: hire a professional programmer, use a special in-house developed high level laboratory oriented language (70-72,74) or the chemist would learn how to do his own programming. The basic problems associated with hiring professional programmers include communication difficulties between the chemist and the programmer concerning the exact requirements of the software and also the basic inflexibility of this approach in dealing with changing requirements of the laboratory. Special in-house laboratory oriented high-level languages never have achieved widespread popularity since the initial effort of developing them for each particular type of computer and interface requires too much time investment. Worse still some of these languages wasted some of the computer's power by not making some functions of the computer accessible to the user (71,72), and also lacked good error diagnostic messages.

A number of developments have made the task of

educating the chemist to be his own programmer easier and more attractive. First, the languages available on small computers have developed far past the original assembly languages so that most general purpose computers will support FORTRAN or BASIC and often times both of these languages. Also the price of peripherals has declined so that mass-storage based operating systems are often available to the chemist. These systems make the computer easier to use and decrease programming time significantly.

Another consequence of the declining cost of computer mainframes has been the increased proportion of the cost of an experiment in terms of both time and money that was invested in software. In fact, in some cases software costs may comprise 80% of the total development costs (74). The availability of good software support is now one of the major factors in the choice of which particular computer a chemist will purchase. Even if the choice of the computer is already made, there is still the choice of what language to use. Writing in a high level language may use 50 to 300% more core memory (75) and slower execution time, but still may be the method of choice because it takes less time to learn the language and less time to write and debug a program. Poor programming techniques and an inadequate programming philosophy in either assembly language or any high level language can severely limit

the capabilities of the interfaced instrument, limit future modifications and expansion, expand software maintenance costs, and needlessly limit the instrument's throughput. The remainder of this chapter describes how these considerations affected the design of the software of an electro-thermal atomizer AA instrument. The resulting package of hardware described in Chapter 2 and this software yielded a versatile, powerful, and easy to use analytical research instrument.

B. CHOICE OF A PROGRAMMING LANGUAGE

To make an intelligent choice of a programming language it is first necessary to define clearly the tasks the program will be required to handle. The minimum a reasonably useful program would be expected to do for our experiments included sequencing the operations of the autosampler, positioner, braid heating apparatus, and data acquisition system. Also, it should compute the integrated absorbance and have provisions for automatically scanning the horizontal and vertical displacements for a one or two dimension profile of the atomic cloud above the braid. Further, a hardcopy of the integrated absorbance and experimental parameters should be available and a plot of absorbance versus time should be presented on the system terminal. In summary, the program is required

to do significantly complex calculations both for determining integrated absorbance and scanning parameters, and also control a significant amount of I/O with the computer's peripherals and the instrument's interface.

The languages available on the 8/e which might be used to write this program included: PALD, PAL8, MACRO8, SABR, RALF, FOCAL, BASIC, FORTRAN II, and FORTRAN IV. Of these PALD, MACRO8, and FOCAL are not supported by the OS/8 operating system and since the operating system is a powerful tool used in creating, debugging, and executing programs, these three languages were immediately rejected as unsuitable. The assembly languages remaining, PAL8, SABR, and RALF, were rejected as possible alternatives for two reasons. First, the program required significant device dependent I/O, computation, and complex sequencing, all of which are much more easily dealt with in a high level programming language. Second, the minimum required program, even when written in a high level language, should fit in the available core memory and not significantly degrade execution time performance.

The remaining high level languages are FORTRAN II, FORTRAN IV and BASIC. FORTRAN IV and BASIC are more powerful than FORTRAN II with respect to the capabilities designed into the compilers. However, each time machine level code needs to be executed it must be referenced as a subroutine call. Writing this code requires a good

deal of familiarity with the subroutine calling conventions of these compilers and also is memory inefficient. On the other hand FORTRAN II allows in line insertion of SABR code which allows the programmer to write efficient code while remaining fairly ignorant of much of the inner workings of the FORTRAN II compiler. Further, FORTRAN II does have all the necessary I/O handling and calculation routines. Thus FORTRAN II was initially chosen as the programming language for the instrument, primarily because of ease of use and programmer familiarity with the language.

C. FORTRAN II PROGRAM CAPABILITIES

The program which was written using FORTRAN II as the source language meets all the minimum requirements necessary for a useful program as described in the previous section. It is written in a modular fashion so that whenever possible each major function is allotted its own particular subroutine. Several of these routines were written by Dr. Eric Johnson including the routines to drive the autosampler (SAMP), positioner (POS), the routine to oversee one sample run (GBRUN), and the plotting package (ADPLOT).

The main program, GBEXEC, sequences calls to subroutines to initialize the positioner, initialize the autosampler, and take the dark current reading. Then it

interacts with the user so that he may choose such things as: the type of experiment (autosampling; zero, one, or two dimensional scan; etc.), and program options (hard copy of results, plot of absorbance versus time on system terminal, etc.). When a requested series of sample runs has been completed the program allows the user several options concerned with changing parameter values, changing program options, or exiting the routine.

For each individual sample run GBEXEC sets up the proper variable values and passes them to a subroutine called GBRUN. This routine interprets the variable values so that the proper subroutines are called in the proper sequence for one complete sample run. This will include some or all of the following functions: sample deposition (SAMP), movement of the braid to the sampling position (POS), heating the braid (TKDATA), and data acquisition (TKDATA), calculation of integrated absorbance, hard copy of results, and plot of results (ADPLOT library).

The TKDATA subroutine sets the digital to analog converter for the proper heating levels for the desolvate, ash, and atomize powers. The routine uses the real time clock (CLOCK) routines to execute the programmed timed delays in the delay, desolvation, and ash steps. The atomize time delay is provided internally to the data acquisition section of the TKDATA routine. This section takes data at 500 Hz and stores the individual analog

to digital conversions in the FORTRAN II defined arrays IDATA and IBACK. The data are routed to IDATA for a sample run and IBACK for a background run.

In addition to the above routines, several utility routines were developed including one named BELL. This routine was used to signal the operator that his interaction was required to do some mechanical function (such as open the PMT shutter) or to alert him that the present task had been completed. The program caused the terminal bell to be rung periodically until the user typed a special character which caused either resumption of program execution or an exit to keyboard monitor (the OS/8 operating system).

The program as described occupied all of the available four fields of core memory. It used the fixed volume sampler designed by Akbar Montaser (29), and the program was used routinely in the process of checking the reproducibility and long-term drift characteristics of this sampler when it was coupled with the graphite braid atomizer.

D. THE PDP 8/e FORTRAN IV SOFTWARE

The FORTRAN II source program described in the previous section was adequate for initial experimentation, but proved somewhat inflexible in meeting the changing demands of our research program. First, the original

autosampler needed to be replaced by one that was more versatile and reproducible. This second sampler required much more complicated machine level coding, which would have further cramped the available memory. Furthermore, any significant departure from the alternatives designed into the program required wrestling with modifying the already tortuous flow diagram of GBEXEC. Writing an equivalent but more powerful program using FORTRAN IV became a necessity. Some of the advantages gained by using FORTRAN IV as a source language were the following: increased effective program size through the use of FORTRAN IV overlaying capabilities; improved calculation times since FORTRAN IV uses the extended arithmetic element - FORTRAN II does not; improved program execution time due to the efficient interrupt driven structure of the FORTRAN IV run time system; DEC's FORTRAN IV also has full ANSI compatibility so that programs from outside sources can be more easily assimilated into the experimental routines; and FORTRAN IV has a more powerful set of possible program statements and instructions, which make programming and debugging easier.

Before attempting to explain the structure and capabilities of the FORTRAN IV GBA program, it is necessary to explain briefly DEC's FORTRAN IV software package. The package contains four major components: a compiler, assembler, loader, and run time system. The FORTRAN IV

compiler produces source files in RALF assembly language in three passes with an optional fourth pass for listing files. The RALF assembler accepts the compiler output and creates relocatable binary modules for execution on machines that do not have a floating point processor (if the FFP is available a FLAP assembler is used). The third component is a LOADER which combines all of the independently compiled and assembled relocatable binary files of the main program, subroutines, and library routines into a load module. The fourth component is the Fortran Run Time System (FRTS). This program supervises loading the load module, swapping overlays as necessary, and handles the I/O requested by the program.

The capabilities of the FORTRAN IV compiler not available in FORTRAN II that are of particular interest for our uses include the availability of logic variables, logical IF statements, and Hollerith formats, which allowed easy implementation of the monitor type program structure described later. In line insertion of error statements in listing files greatly simplifies debugging operations. Most important, however, was the possibility of more versatile device independent I/O, which allowed hard copy results to be generated on the line printer and data to be stored in two or more data files open for input concurrently. FORTRAN IV also has available an extensive library of utility routines. One utility routine of

particular note is ONQI which adds the user written interrupt handler to the run time system's polled interrupt skip chain. Another utility routine, ONQB, adds a user written routine to the background task list of the run time system. These two routines allow the programmer to structure execution of routines such as plotting and device drivers (e.g., autosampler handler) in such a manner that instrument throughput is enhanced. Compiler options A and F which allow the generation of RALF assembly language source files, and fully annoted listing files, respectively, are of great utility. Thus, when machine level functions are required, a skeleton FORTRAN subroutine can be compiled under the A option and can then be edited so that the desired machine level code is added at the proper position. This greatly simplifies the machine level subroutine writing, since much of the calling convention and argument passing is already taken care of by the compiler. This modified routine can then be loaded with FORTRAN routines and called in the same manner as any other FORTRAN subroutine call. To modify a RALF source file, the programmer need only understand a few of the RALF instructions and argument passing conventions. The RALF assembler operates in two modes: the ralf mode and the eight mode. The ralf mode has one, two, or three word instructions that deal with a software simulated three word floating point accumulator and seven, one word

index registers. This format allows 15 bit direct addressing, base page relative addressing, and register indirect addressing. The eight-mode RALF code uses the 8/e machine level instruction set of PAL8 except for a slightly different indirect addressing convention and a different method to indicate the memory locations occupied by the routine. The main task of the machine level programmer is to conform to the RALF conventions with respect to argument passing and transferring execution back and forth from the ralf-mode and the eight-mode code.

The FORTRAN IV loader differs from the FORTRAN II loader in that it allows overlaying. Up to eight levels of overlays are allowed with up to 16 overlays per level, except for level 0 which is always core resident and may never be overlayed. Level 0 must contain the main program, library routines, and any modules containing eight-mode RALF code. There are a few important restrictions to keep in mind when designing an overlay structure. Functionally, the loader regards all overlays in a given level as having the same length as the longest overlay. All overlays are multiples of two blocks long. Also, as a general rule, no program may execute a call which would cause itself or any program that called it to be overlayed. The loader provides an optional loader map listing which designates overlay levels, program locations, etc. In the loader map there are two error messages which do not

appear in the DEC documentation. An * beside a subroutine name designates that there is a possible violation of the overlay calling rule mentioned above. A *EX designates that the loader did not locate any entry point corresponding to this subroutine name, even though it had been requested to be loaded.

The load module created by the loader is executed under the auspices of the FORTRAN Run Time System (FRTS). It is this program that allows devices to be defined for device independent I/O, sets up appropriate buffers, and provides the interrupt driven handlers to allow foreground I/O with background computation. Another nice feature is error trace back at run time. This function detects fatal run time errors, gives terse error statements, and fingers the culprit both as to line number and subroutine name. Typing a CONTROL B at run time provides this same trace back function.

The PDP 8/e FORTRAN IV package as described here possesses many attributes not found in the FORTRAN II package and was used to generate a more flexible, powerful, and easily modifiable program than the program which was written using FORTRAN II as a source language.

E. PROGRAM STRUCTURE FOR THE EAAA INSTRUMENT

Because of the limitations of the FORTRAN II program and the extended possibilities of programming using the DEC FORTRAN IV package, it was advantageous to expand the tasks required of the GBA software. Rewriting and restructuring the software using FORTRAN IV as the source language resulted in improvements in the following areas: decreased execution time, raw and processed data available in output files, more error checking functions, increased modularity so that software was more easily modified, and added features which expanded the abilities of the program to handle situations and experiments not originally designed into the software.

The overlay structure of the resulting program (GBMDS2) is shown in Figure 10. In the figure, routines which occupy the same level but different overlays are separated by a colon. These routines occupy the same portion of core but are swapped into memory at different times during execution. If the routines occupy the same level and overlay, their names are separated by a comma. These routines are always co-resident in core since the run time system always swaps into core entire overlays not just portions of the requested overlay. The lines connecting programs denote that the program in the upper level may call the lower level routine. Note that routines in level 0 may be accessed by routines in any level, but

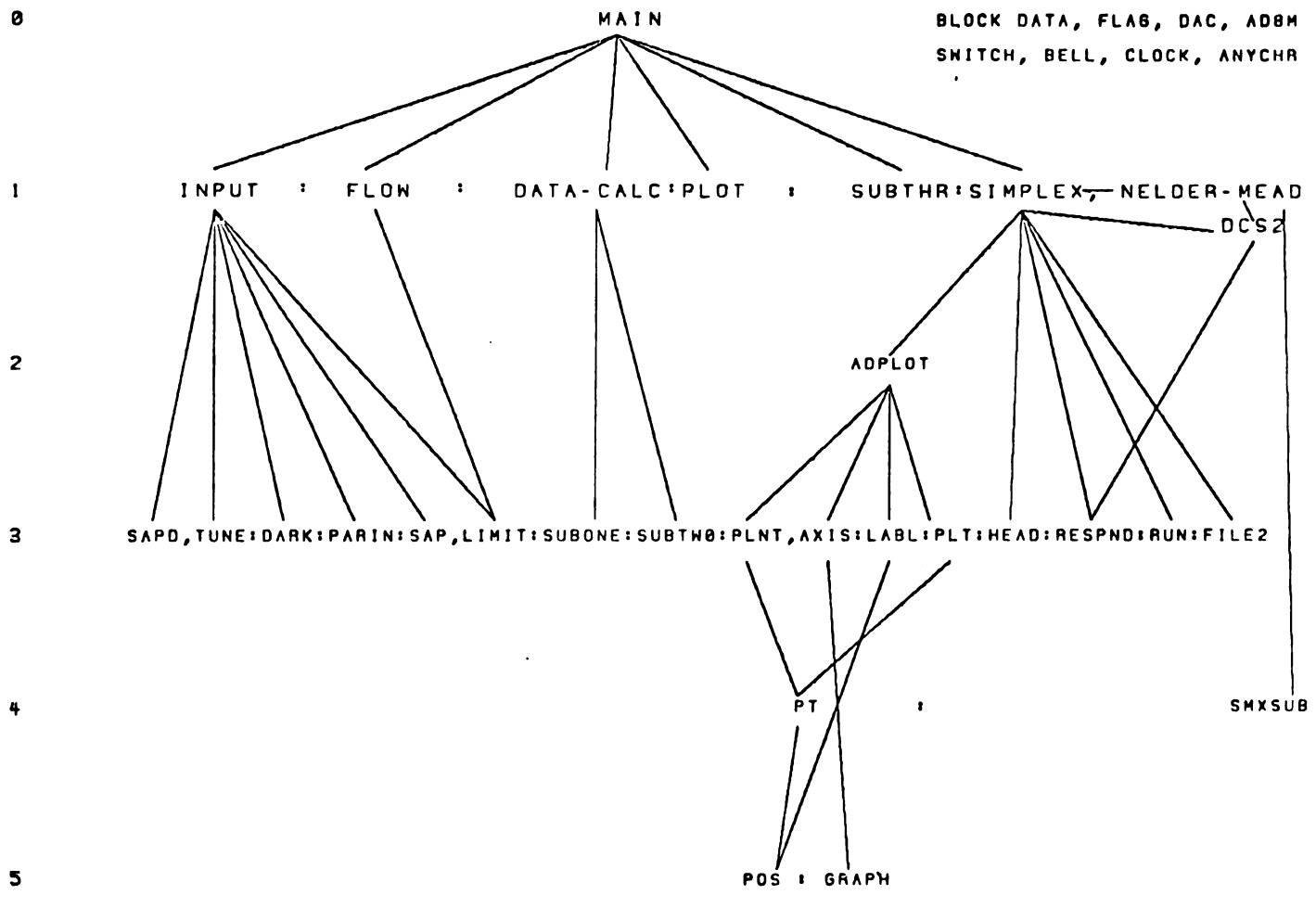


Figure 10. Diagram of the Overlay Structure of the GBMDS2 Program

to avoid mass confusion in the diagram these connecting lines were left out. The routines residing in level 0 are the main routine and the RALF subroutines containing eight-mode code. Since the level 0 routines may not be overlayed, great effort was made to keep these routines as short as possible. Also in level 0 is the special function routine BLOCK DATA, which contains no executable code and is used to initialize the values of common variables.

Subroutine INPUT is called immediately upon starting execution of the main program. Almost all user interaction with the experiment is through this routine. Under user commands the default values of experimental parameters set in the BLOCK DATA subroutine may be changed. Also, initialization and demonstration routines for the positioner, autosampler, and data acquisition system may be executed. When the user completes this setup, he gives a special command to start the execution of the experiment. The control is then passed back to MAIN which normally calls the subroutine FLOW, unless in the simplex mode in which case the SIMPLEX subroutine is called. In the FLOW subroutine, all experimental parameters are checked for illegal values. Error messages are printed for each illegal value detected, and the control is then directed back to subroutine INPUT. If no illegal values were detected, new arguments which will later be passed to the sampler and positioner subroutines are calculated,

the sampler and positioner subroutines are calculated, and control is passed to the DATA-CALC subroutine. The DATA-CALC routine interacts with the sampler, positioner, digital to analog converter (DAC), and analog to digital converter (ADC) subroutines to acquire, calculate, and output the results of a single sample run to the requested output devices. If a plot of absorbance versus time for a sample run was requested, control is passed to the PLOT subroutine. If no plot was requested, control is passed back through the main program to the FLOW subroutine. After the PLOT subroutine has completed a plot of absorbance versus time, program control is passed back through MAIN to the FLOW subroutine, unless the wait mode has been asserted. In the wait mode, the user has the option of requesting a plot of background levels or exiting this routine back to MAIN, which will respond by recalling subroutine FLOW. In overlay five of level one, the simplex package is loaded. Discussion of the functions of these routines has been reserved for Chapter 5.

The complete program GBMDS2 may be easily created if the relocatable sources of the programs in Figure 10 and the FORTRAN IV library are available. This is the first point of easy program modification since any of the individual modules may be modified and used in place of the original version at this point. Further the routines SUBONE, SUBTWO, SUBTHR, and SMXSUB are optional routines

which may, but need not, be loaded concurrently with the original routines. If loaded, they may be executed at specific times during program execution if the user requests them at run time. These routines allow for such things as non-standard calculation schemes or file output formats.

When the user has created one of these new routines or modified one of the system routines, he can create the new GBMDS2 program and start execution using the following dialog. This dialog utilizes the batch stream program GBMDS2.BI to create the new program.

```
.SUBMIT SYS:GBMDS2.BI(UTE)
.R FRTS
*GBMDS2
*FILE1.DA</5
*SYS:FILEZ.DA</6
*$BATCH.TX/4
*$
```

The first line of the dialog starts the batch stream and thus creates the load module. The next line starts the FORTRAN run time system. The first argument to the run time system is the name of the program's load module that has just been created by the batch file. The remaining arguments deal with file structured I/O and will be explained later in the chapter. A more detailed

description of some of the most important subroutines is given in the remainder of this chapter.

F. THE INPUT SUBROUTINE

Most user interaction with the program occurs in the subroutine or subroutines called by INPUT. The structure and capabilities of this routine are probably the most important factors in making the FORTRAN IV program more usable than the FORTRAN II program. An example, which illustrates the default parameter values and a sample dialog is shown in Figure 11. INPUT first gives the present version number which indicates to the user that this is the standard software package or that it contains some special purpose modifications. The program prompts the user to enter a command by outputing a colon and ringing the terminal bell once. At this point the user may enter any one of approximately 80 legal commands. These commands are from two to six alphanumeric characters long. Some of these commands additionally request a numeric argument. A complete list of the commands along with a brief description of the function of each one is given in the documentation section of the INPUT subroutine listing in Appendix A. In the example given in Figure 11, the first command selected by the user was HL. The program responded by giving a listing of the default values of all pertinent variables. The next command reflected the user's desire

```
GBA VER  3.01

:HL

EXP: NO TITLE                                                    6/21/77
PLT F   LPT T   RAD T   HDF F   WAIT F   SCAN F   SAMP F
SB1 F   SB2 F   SB3 F   FL1 T   FL2  F   SINT F   PINT F DARK F
DELT=   10.00   DEST=   10.00   ASHT=     5.00    ATMT=    0.35
DESP=   10.00   ASHP=   0.100   ATMP=     0.500   FREQ= 1000.0
VER=   2.0000   HOR=   0.0000   AMNT=      2.00   NMRUN=      1

:NRAD

:DELT
=20
 .
 .
 .

:HL

EXP: EXAMPLE OF INPUT COMMAND USAGE                              6/21/77
PLT F   LPT T   RAD F   HDF F   WAIT F   SCAN F   SAMP F
SB1 F   SB2 F   SB3 F   FL1 T   FL2  F   SINT F   PINT F DARK F
DELT=   20.00   DEST=   10.00   ASHT=     5.00    ATMT=    0.35
DESP=   10.00   ASHP=   0.100   ATMP=     0.500   FREQ= 1000.0
VER=   2.0000   HOR=   0.0000   AMNT=      2.00   NMRUN=      1
 .
 .
 .

:GO
```

Figure 11. Example of a User Dialog with Subroutine INPUT

to use power programming rather than radiation programming. With the DELT command and the associated numeric argument, the user has changed the time that the instrument is requested to wait between sample runs. Finally, the user checked the status with another HL command and after a few other intervening commands started the experiment with the GO command. In most cases the command to change an argument is the same as mnemonic given in the table of variable status output on an HL command. Some additional commands are worthy of special note. For example, several commands are used to open the data files specified as arguments to the FRTS in the previous section (FILE1.DA and FILE2.DA) so that these files contain the present date, program version number, user comments, etc.. Another set of commands initializes the positioner (PSIN) and the sampler (SPIN). The commands POSDEM and SAMPDM allow the user to test the individual functions of the positioner and sampler. The DARK command takes 100 dark current readings and returns the average. Other commands allow the user to activate calls to the special subroutines he may have written and added to the program.

A special set of commands is necessary to allow the powerful technique of batch streaming command strings from an OS/8 file to the INPUT subroutine. This batch file is created using either EDITOR or TECO to save a list of commands that the user creates just as if he

were entering them directly to the subroutine INPUT from the system terminal. This file may then be specified to FRTS as device four (file BATCH.TX in the example given in the previous section). When the INPUT subroutine is executed, it requests commands from the system terminal until it receives the command BATCH. Then commands are requested from the previously designated batch file and echoed at the system terminal until either an erroneous command or the command LOCAL is detected in the batch file. At this point commands are again requested at the system terminal. Subsequent BATCH commands will start command execution from the batch file at the command immediately following the last one executed from the file in the last access. A few functions accessible by INPUT commands require user interaction. For example, to take the dark current, the user must be prompted to close the PMT shutter, and program execution must wait until this function has been completed. Two special batch-mode commands help handle this situation. The MSG command causes the next line of code in the batch file to be echoed to the system terminal and not interpreted as a command. In this case the message might be "MASTER PLEASE CLOSE THE PMT SHUTTER SO THAT WE CAN CONTINUE THE EXPERIMENT". The BELL command is then used to ring the terminal bell periodically until the user completes the task and exits the bell routine by typing a character at the system

terminal. All programs that require some run time user interaction similar to the previous example always request some interaction at the terminal, even when in the batch mode, so that these functions are never accidently bypassed.

Uses of this batch mode of operation are numerous. The user could easily create a library of batch files under each element's name and each of these would contain the proper experimental parameters for that particular element, along with any auxiliary messages to the user concerning any necessary special procedures. In more complex applications, the user might use the batch stream to execute a series of tasks not specifically provided for in the basic software. For example, the user might wish to scan through various atomization powers; a task which is not easily done with the normal routine. However, with the appropriate batch stream this non-standard experiment could be executed without the user having to interact with the program personally each time he wished to change that parameter. An adequately constructed batch file should allow even a moderately competent technician to operate the instrument with good results.

A simplified flow diagram for the INPUT routine is given in Figure 12. Note that this type of free-flowing structure makes it quite a simple task to insert or delete

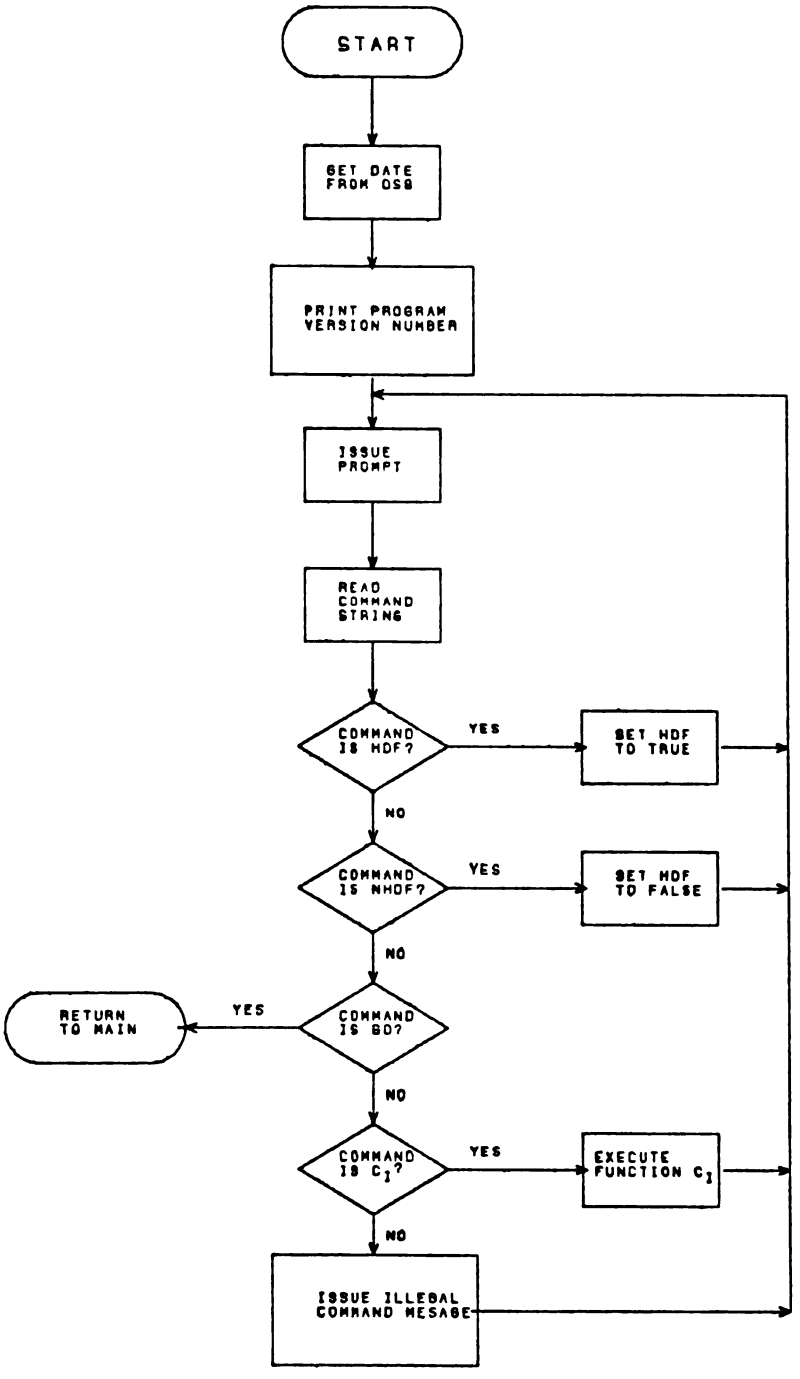


Figure 12. Simplified Flow Chart for Subroutine INPUT

commands, since the position at which the command check is inserted is quite arbitrary (it only needs to be inserted somewhere amongst the existing command checks).

In summary, the main attributes of the subroutine INPUT include its ability to use batch-mode command input to make the instrument simple, direct, flexible and highly automated. Also, the loosely structured program construction allows easy modification such as deleting or adding commands as needed. Demonstration routines for the sampler and positioner are useful in equipment testing and maintenance operations. The file structure and optional user subroutines add to the versatility of the program and allow it to meet the changing demands of a research project.

G. THE FLOW SUBROUTINE

As stated previously, the FLOW subroutine handles the generation of the proper variable values for a sample run when the experiment involves either position scanning or automatic sample deposition. Since some of the modes of operation are quite complex, the flow chart for this subroutine is not included here. For example, a two dimensional scan proceeds by starting at the requested position and scanning vertically at this position and then repeating this vertical scan at horizontal increments first at a positive displacement and then at an equal magnitude

negative displacement, then at a larger positive displacement, etc. Superimposed on this action may be requests for a series of different samples and sample sizes. Further options allow a number of sample runs to be taken at each scan point and more than one scan to be taken. This routine also has the assignment of detecting the end of the assigned task and passing control through MAIN back to the INPUT subroutine, rather than the normal mode where control is passed to the DATA-CALC routine. At each call this routine scans a hardware switch register which may be used for several functions. An end of task can be simulated by setting one of these switches so that control is immediately passed back to INPUT. Several other switches are used to output the status of run counters, scan variables, and sampler contents. Another switch forces a call to PLOT, which produces an absorbance versus time plot of the previous sample run. These switch register options are very important in program debugging and checking the status of a long term experiment, but since these tasks are implemented only on request system throughput is not degraded. The program listing for FLOW is given in Appendix A.

H. THE DATA-CALC AND PLOT SUBROUTINES

The DATA-CALC subroutine is the routine that interacts with the instrument's hardware in a typical sample run. The sequence involved in a sample run is diagrammed in Figure 13. On entry this routine first checks to see if the positioner is presently busy, and if not busy, it is sent to the sample pick up position. If the required time has elapsed since the last sample run, the autosampler is engaged, and the requested sample is deposited on the braid. The positioner is sent to the requested test location, and the braid is heated at the desolvate power for the desolvate time. Now the 100% T reading is taken. The reasons for waiting until this time to take the 100% T reading are: first, if the sample has not been desolvated the sample droplet may occlude some of the light; second, the 100% T reading varies at different sample locations due to changes in the optical properties of the quartz cell walls; finally, to minimize the effects of HCDT drift, the reading is taken as close to the atomization time as possible. Following this the ash step is executed, and then the radiation and hydrogen diffusion flame flag is set if these were requested for the atomization step. During the sample run the transmittance data are stored in the array XVAL. Following this an abbreviated sequence of delay, ash, and atomize steps is carried out to acquire and store data related to the background

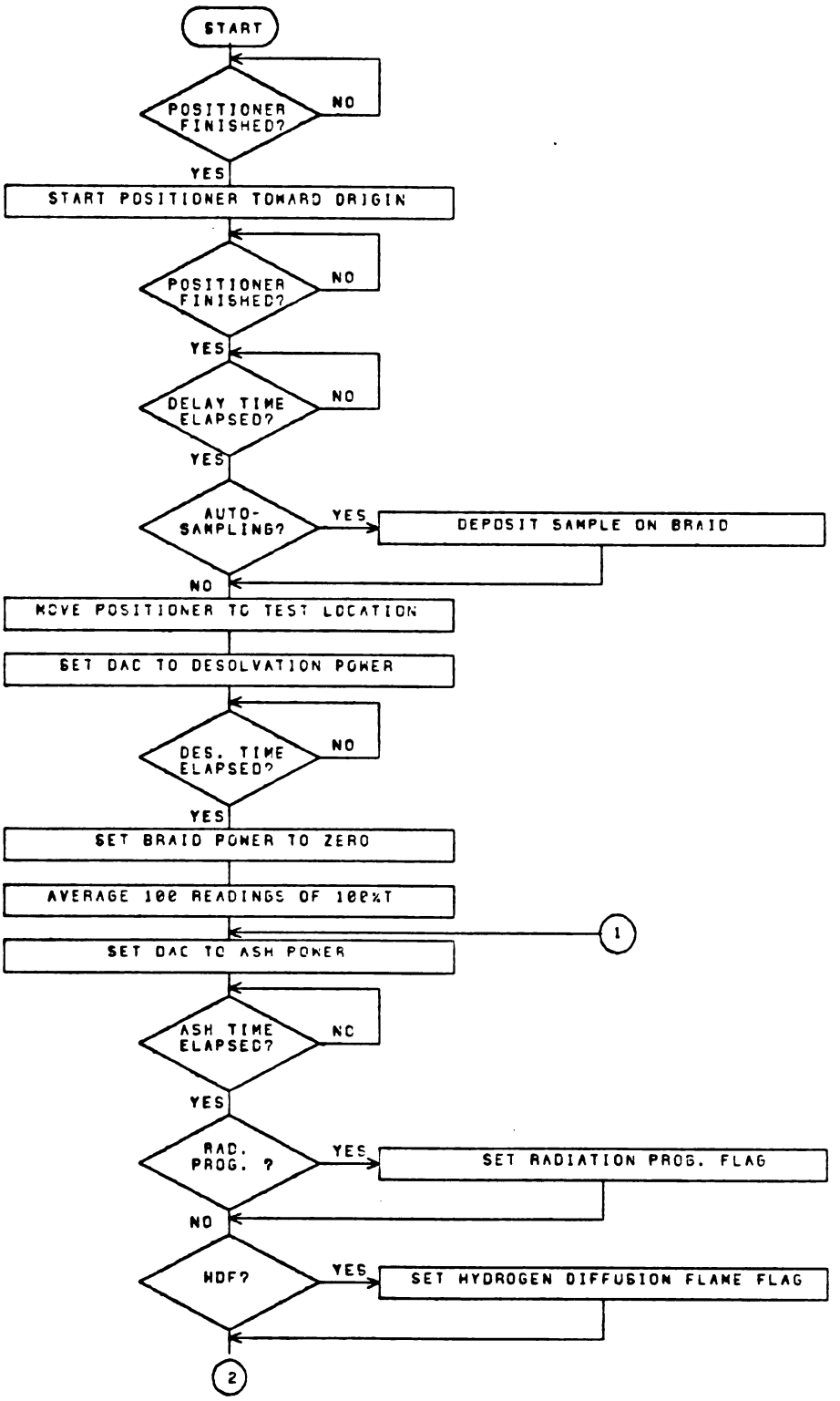


Figure 13. Flow Diagram for the DATA-CALC Subroutine.

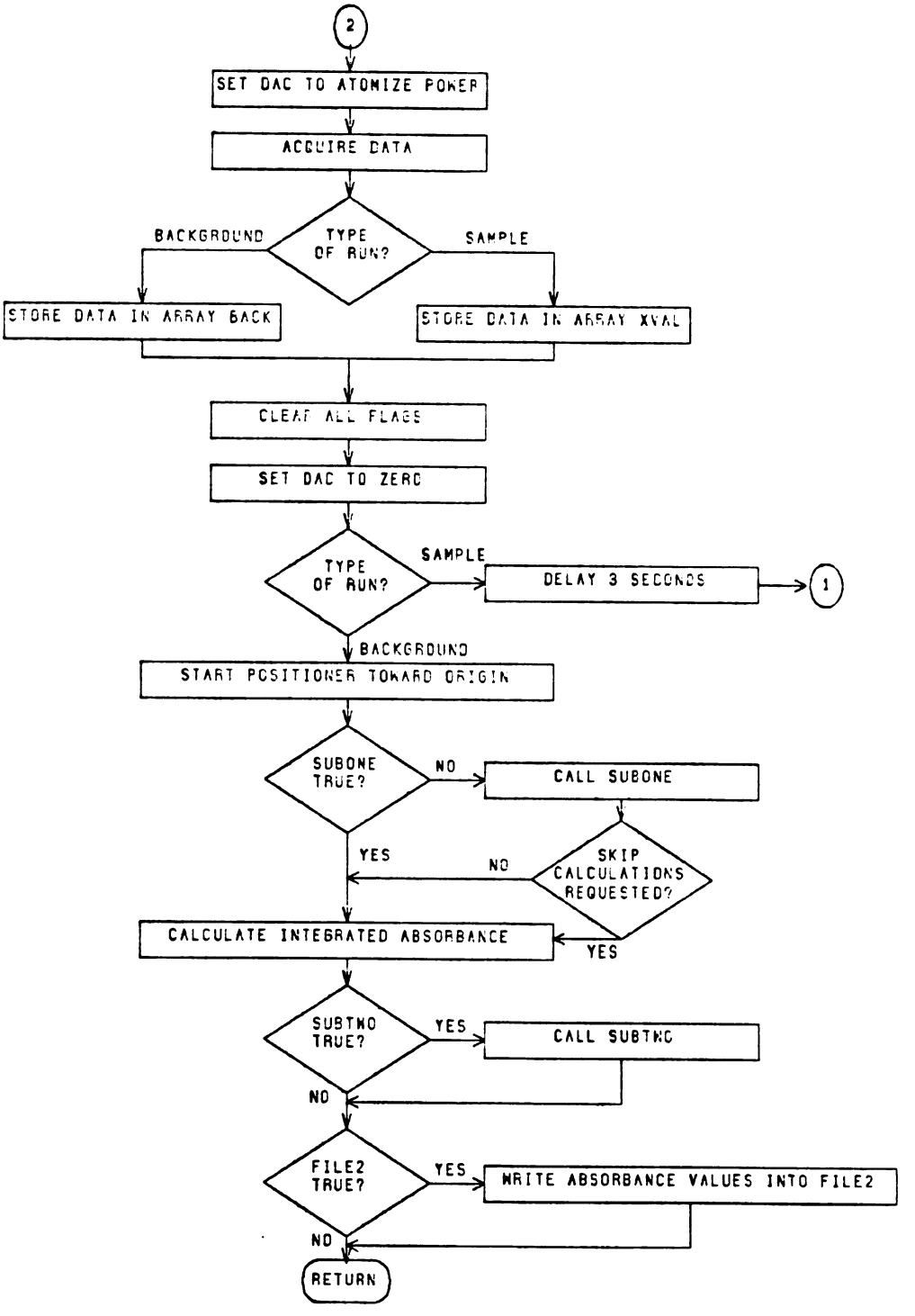


Figure 13. Flow Diagram of the DATA-CALC Subroutine (continued).

emission of the braid. The positioner is then started back toward the sample deposition position, and the calculation of the integrated absorbance is begun.

The calculation of the integrated absorbance for a sample run requires four measurements: 100% T, sample absorbance, background braid emission, and dark current. The photocurrent components of these are given below.

$$I_1 = I_{OL} + I_D$$
$$I_2 = I_{AL} + I_D + I_{BE} + I_{ME}$$
$$I_3 = I_D + I_{BE} + I_{OL}$$
$$I_4 = I_D$$

where I_{OL} is the photocurrent due to the unattenuated line source, I_D is the dark current, I_{BE} is photocurrent due to continuum braid emission at the analysis wavelength, I_{ME} is the photocurrent due to the thermally excited atomic emission of the analyte, and I_{AL} is the photocurrent due to the line source when it has been attenuated by the analyte and any background absorbance of the matrix at the analysis wavelength. If the contributions to the photocurrent by analyte emission and continuum background absorbance are negligible then the absorbance at any given time is

$$A_L = \log \frac{I_{OL}}{I_L} = \log \frac{I_1 - I_4}{I_2 - I_3 + I_1 - I_4} \quad ,$$

and the integrated absorbance is the sum of these absorbances over the specified time interval. Note that this is not a true integration and that it depends on sampling frequency and peak shape (28).

In the actual coding, the mathematical identity that states that the sum of a series of logarithms is equal to the log of the multiplicative product of the series was used. This improves execution speed since multiplication is a much faster computer operation than computing logarithms.

If a continuum source is used instead of the line source, the pertinent measurements are

$$I_4 = I_D$$
$$I_5 = I_{OE} + I_D$$
$$I_6 = I_E + I_{BE} + I_D + I_{ME}$$
$$I_7 = I_D + I_{BE} + I_{OE}$$

The new photocurrent components are I_{OC}, which is the photocurrent from the unattenuated continuum source, and I_H, which is the attenuated continuum source. Because of the bandpass of the monochromator, the attenuation of the continuum source by the analyte can be disregarded when compared to any broad band absorbance by matrix species. Again if the analyte emission is neglected

$$A_c = \log \frac{I_{OC}}{I_C} = \log \frac{I_5 - I_4}{I_6 - I_7 + I_5 - I_4}$$

If the background level is constant from run to run, the true sample absorbance (A_s) is $A_L = A_S - A_c$. For most of our work A_c was assumed to be very small so that A_L was taken to approximate A_S.

The PLOT subroutine uses a plotting package written by Dr. E. R. Johnson to present an absorbance versus time plot. In this plot, the origin of the time axis is the time at which atomization power was first applied to the braid. When the plot has been completed, and if not in the wait mode, the program control is directed back to the subroutine FLOW. If in the wait mode the user may request a plot of background transmittance versus time. Normally, however, the wait mode is used to allow manual sample deposition, and program execution may be restarted on command when this manual deposition has been completed.

Listings of the DATA-CALC and PLOT subroutines are in Appendix A, and more information concerning the exact mode of operation can be extracted from the extensive documentation at the beginning of each listing. The important features to note in the DATA-CALC routine are the switch register options, which aided debugging and progress monitoring of long running experiments. Also, note the importance of point by point braid emission background correction of absorbance values since especially in the beginning of the atomization period this background level changes rapidly.

I. ASSEMBLY LANGUAGE SUBROUTINES

Because of the number of hardware devices that needed to be controlled in the GBA instrument, a good deal of assembly language programming was necessary. In some cases the skeletons of these RALF routines were derived from compiling FORTRAN IV subroutines as suggested early in this chapter. In other cases where no argument passing, etc., was necessary the FORTRAN IV compiler was overly verbose so that even the ralf-mode RALF code was written without aid from the compiler. The program ADCR is an example of this type and its listing is given in Appendix A. Also in Appendix A is the listing of AD8M which is the eight-mode complement of ADCR. These two routines function together to acquire data, convert the data to FORTRAN IV compatibility, and store the data in a floating point array in the calling program. To use this routine, three programs in the ADCR package must be called. First, FREQS is called with an argument denoting the requested frequency of conversion. The requested frequency is rounded off to a multiple of 1., 2., or 5. between $1. \times 10^{-3}$ and $5. \times 10^{4}$. Then the proper arguments are calculated, passed to the GBA hardware clock, and the clock started.

The user then calls SET to pass the number of conversions requested and the name of the data array. Finally, when the atomization step is to begin, the user

calls GO to initiate the actual data taking. The ralf-mode programs FREQS, SET, and GO interface to the respective complementary sections of eight mode code in AD8M. The eight-mode code and ralf-mode code were kept separate so that the ralf-mode routines could be overlayed if necessary.

The complex portions of the task are handled mostly by the eight-mode code. First it must pick up the address of the data array in the calling program and determine the data field in which it starts. As data are acquired, they are converted to FORTRAN IV floating variable form using the Extended Arithematic Element and stored in the requested data array. Another level of complexity is added by the fact that FORTRAN IV arrays may be loaded across field boundaries so that the code must also detect when a data field change is necessary. The routine will execute without missing a data point at data rates in excess of one kilohertz. If, however, the requested conversion frequency is excessive and causes a data point to be missed, an error halt with 1 in the accumulator occurs.

A single FORTRAN IV callable RALF subroutine handles all the functions of the positioner:

CALL POS (HPOS, VPOS, COM)

where HPOS is the horizontal destination; VPOS is the vertical destination; and COM is the command type. There

are three possible command types: RESET, WAIT, and GO. The RESET command causes the positioner to seek the reference position defined by the vertical and horizontal optical interrupters. The WAIT command causes the computer to wait for the positioner to set its done flag before returning from this subroutine to the calling routine.

The remote mode GO command functions much the same as the local mode GO command with a few minor distinctions. The horizontal and vertical destinations are the HPOS and VPOS subroutine arguments, respectively. The software converts the arguments to the machine coordinate system (which is 12-bit binary with the 0,0 position at the optically defined reference point) from the user coordinate system, which uses mm as units and has the origin defined as the position where the atomizer is just below the HCDT light path (i.e., midway horizontal travel, vertical as high as possible without blocking the light beam). Thus legal destinations are from -12.5 to 12.5 mm horizontal and from 0. to 25.0 mm vertical.

A FORTRAN IV callable subroutine written in RALF is used to drive the DAC. The calling form is:

```
CALL DAC (ARG)
```

where ARG is a real number which represents the desired output voltage. The unique feature of this subroutine

is that it may be compiled under several different compiler options, which can cause the argument to be interpreted differently (69). If no compiler option is specified at compilation time, the real argument is converted to its binary equivalent and passed directly to the DAC input latch. With compiler option 1, the DAC is assumed to be a unipolar 0. to 10. V DAC, and the argument will be interpreted as a voltage. This voltage will appear at the DAC output. Compiler option 5 produces a similar interpretation. But in this case, the DAC is assumed to be a bipolar -5 to +5 V DAC.

Subroutine FLAG is also written in RALF assembly language and is called in the following manner:

```
CALL FLAG (RAD, HD, SP)
```

where RAD, HD, and SP are real arguments with the values of -1, 0, or +1. An argument with a value of +1 causes the flag to be set. A -1 argument causes the flag to be cleared, and 0 does not affect the state of the flag. The RAD argument is used to set or clear the flag, which causes a changover to radiation regulation of the atomizer heating. The HD argument is used to set or clear the flag which triggers the gas valve for the hydrogen diffusion flame. The SP flag operates similarly to the RAD and HD flags, but it has not yet been assigned a specific purpose.

The subroutine which reads the user switch register into the accumulator is actually a function type subroutine and an example of a typical call to this routine is given below.

```
IF (SWITCH (5.) .EQ. 1.) CALL PLOT
```

The argument to the function subroutine indicates which switch is being interrogated, and, if that switch is set, the function returns a value of 1.. If it were not set, a value of 0. is returned. In this example, the SWITCH function subroutine is used in a logical IF statement so that the PLOT subroutine is called only if switch 5 is set. The functions actuated when the other switches are set are given in Table 2.

Although this switch register may at first seem to serve a rather frivolous function, in actual practice it is almost indispensible. In fact, the necessity for this type of interaction was not recognized in the original design. However, it became so necessary to have this type of run-time control, this switch register was added after the construction and testing of the rest of the circuitry was completed.

Also used by GMBDS2 are the clock routines written by Dr. E. R. Johnson. These routines use the 8/e's real time clock to provide accurate time delays. The software to drive the automatic sample dispenser is discussed in some detail in Chapter 4 and will not be discussed here.

Table 2. Run-time Front Panel Switch Options.

Switch	Function
0	Interrupt task execution, go to GBA monitor
1	Print run counter, position, and logic variables on system terminal
2	Print scan variables on system terminal
3	Print sample dispenser variables on system terminal
4	Plot Absorbance vs. Time on system terminal
5	Unassigned

CHAPTER 4

CONSTRUCTION AND TESTING OF THE VARIABLE VOLUME AUTOMATIC SAMPLE DISPENSER

A. INTRODUCTION

Automatic sample delivery in electrothermal atomization atomic absorption (EAAA) measurements can improve measurement precision and result in a substantial time savings for the experimenter. Mechanically assisted (76), semi-automatic (77,78), and automatic (68,79,80), discontinuous sample introduction systems have shown that these types of systems can improve both the precision of volume metering and the precision of positioning the sample over manual sample delivery. In many cases however, the gain in precision is much less important than the possible time savings. A great number of the AA instruments equipped with electrothermal atomizers have automated all instrument functions except sample introduction, so that while the instrument handles all the rigorous tasks such as data acquisition and analysis the experimenter is left with the mundane task of operating a hypodermic syringe.

In our laboratory we wished to perform experiments to

map the atomic vapor in two dimensions above the atomizer. Each of these maps would require at least one hundred sample depositions and such maps would be required for several elements, matrices, and atomization temperatures. With a task of this magnitude, automation of the sample introduction technique was imperative. Choice of the particular design of the sample introduction system was affected by previous experience with automatic sample delivery systems (79,80) and the physical constraint that the system had to fit on an optical rail already crowded by the presence of the automatic positioner for the atomizer cell (28). Further, to yield maximum flexibility the system was designed so that the sample volume and the sample type could be changed under computer control.

The relationship of the Automatic Sample Dispenser (ASD) to the other components of the computer-controlled EAAA instrument is shown in Figure 14. The diagram has been simplified for clarity by omitting the automatic atomizer vapor cell positioning device. The ASD consists of a sample turntable module, a transport and sample delivery module, and the electronics associated with controlling these modules. The sample turntable module contains sample cups which allow the selection of any of six different samples. The transport mechanism provides vertical movement to lower the delivery system either into the selected cup or onto the filament atomizer. The rotor

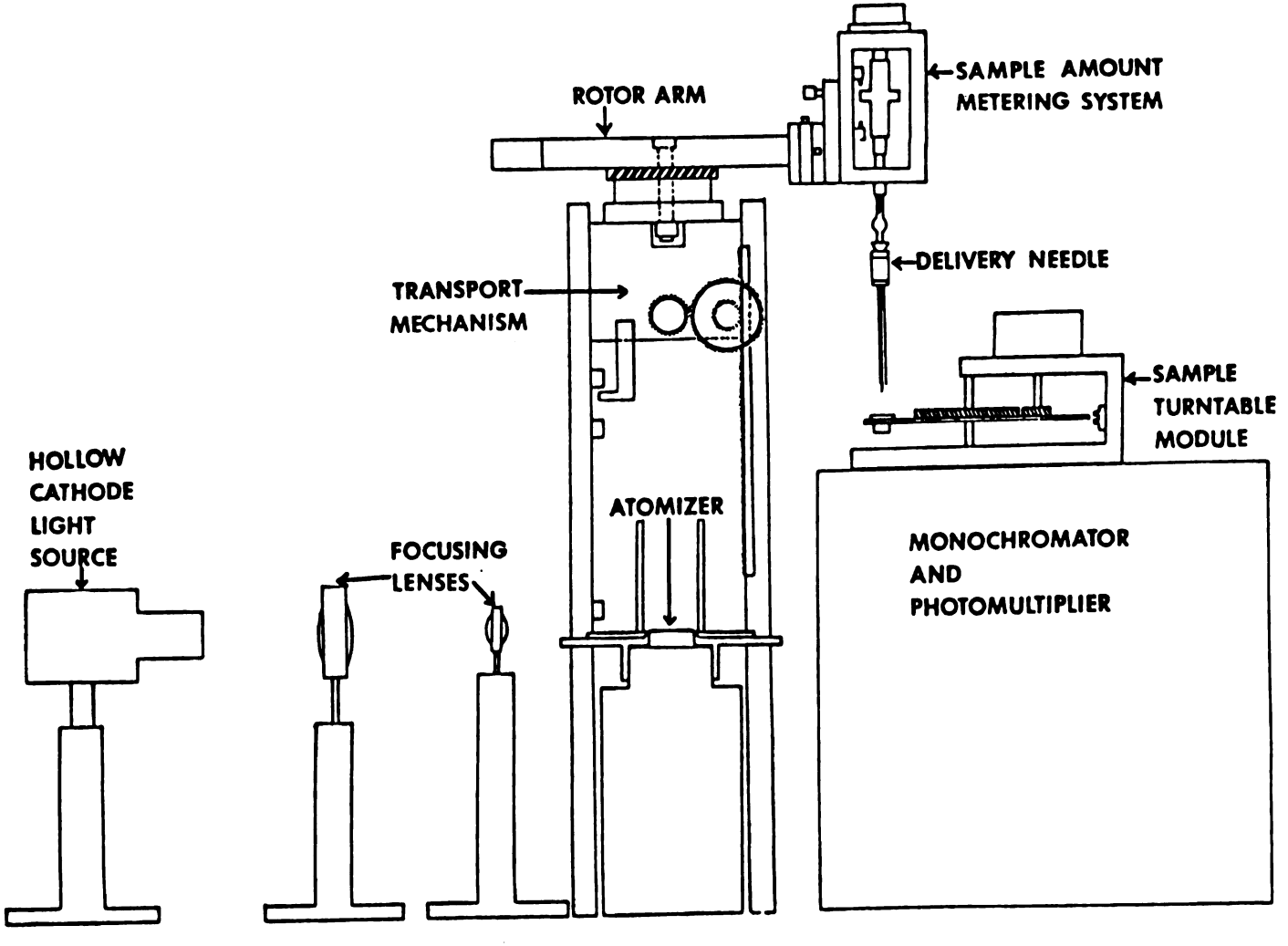


Figure 14. EAAA Instrument with Automatic Sample Dispenser

arm rotates the delivery system between the position where the delivery needle is directly above the sample turntable and the position where the needle is directly above the center of the atomizer.

B. CONSTRUCTION AND OPERATION OF THE ASD

1. Mechanical Components

The components of the automatic sample dispenser (ASD) are shown in Figures 15, 16 and 17. The delivery system is shown in detail in Figure 15 and is based on the adaptation of a 200 µℓ manual syringe (S1100, Gilmont Instruments, Inc., Great Neck, N.Y.) for automatic operation. When stepper motor A is pulsed, the coupling adaptor C and outer syringe sleeve D rotate with the motor shaft since the outer syringe sleeve and the syringe body are threaded, motor rotation causes the syringe sleeve to advance or retreat on the syringe body. The teflon syringe plunger F is mechanically coupled to the outer syringe sleeve D and so advances or retreats inside the fixed position glass barrel G whenever the motor is pulsed. An air tight seal is maintained by a teflon plate and Vitron O-ring inside nut E. Thus, motor movements cause the displacement of whatever fluid is contained in the glass barrel. Teflon high pressure liquid chromatography components H couple the .3 mm i.d. teflon tubing to the

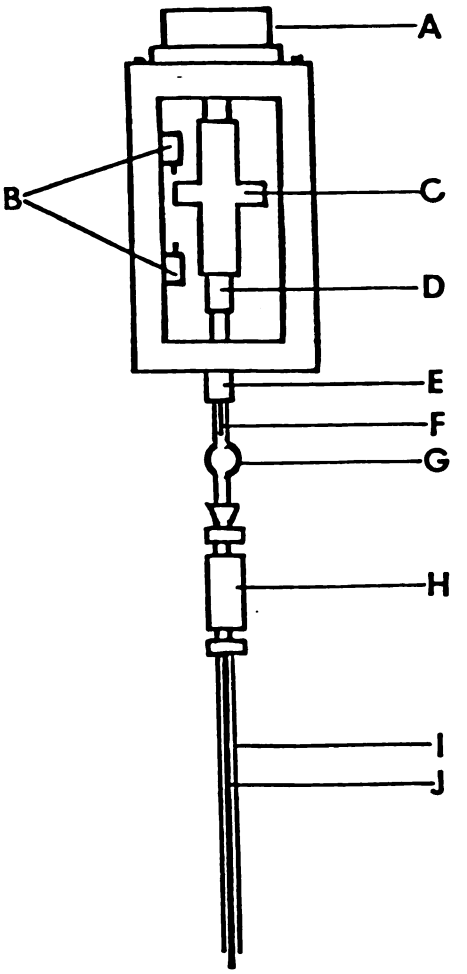
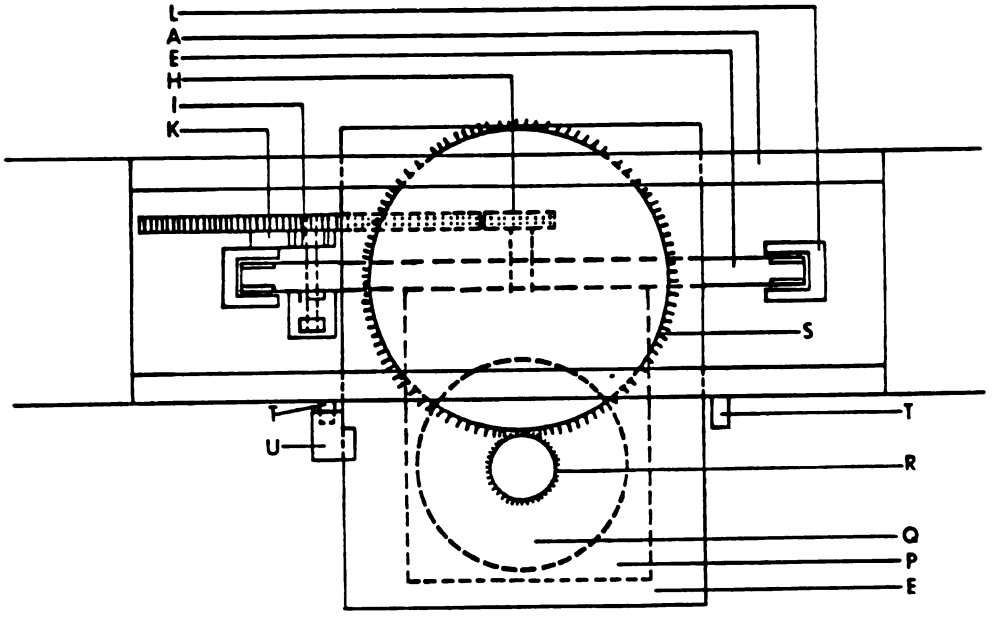


Figure 15. Sample Amount Metering and Delivery System

(A) stepper motor, (B) limit microswitches, (C) coupling adaptor, (D) outer syringe sleeve, (E) plastic nut, (F) Teflon syringe plunger, (G) precision bore glass barrel, (H) HPLC components, (I) glass tubing, (J) Teflon tubing

Figure 16. Transport Mechanism.
(A) stepper motor, (B) counter weight, (C) 2-1/2" gear, (D) Teflon spacer (E&F) aluminum transport plates, (G) occluder arm for interrupters, (H) 1/2" gear, (I) 1/2", 2-1/2" double gear, (J) optical interrupters, (K) 11" long geared rack, (L) channeled aluminum struts, (M) base plate, (N) delivery system and amount metering system, (O) dovetail slide system for x, y, z positioning, (P) vertical motor, (Q) rotation motor, (R) 1/2" gear, (T) optical occluders, (U) optical interrupters.



TOP VIEW

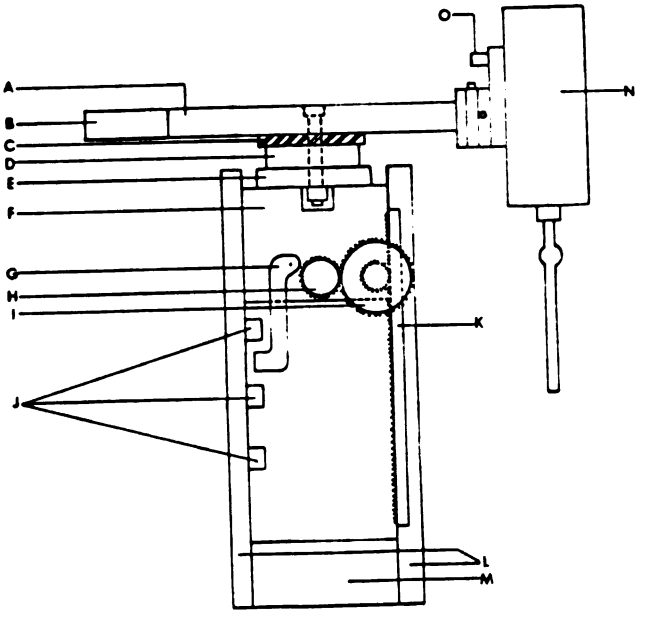


Figure 16.

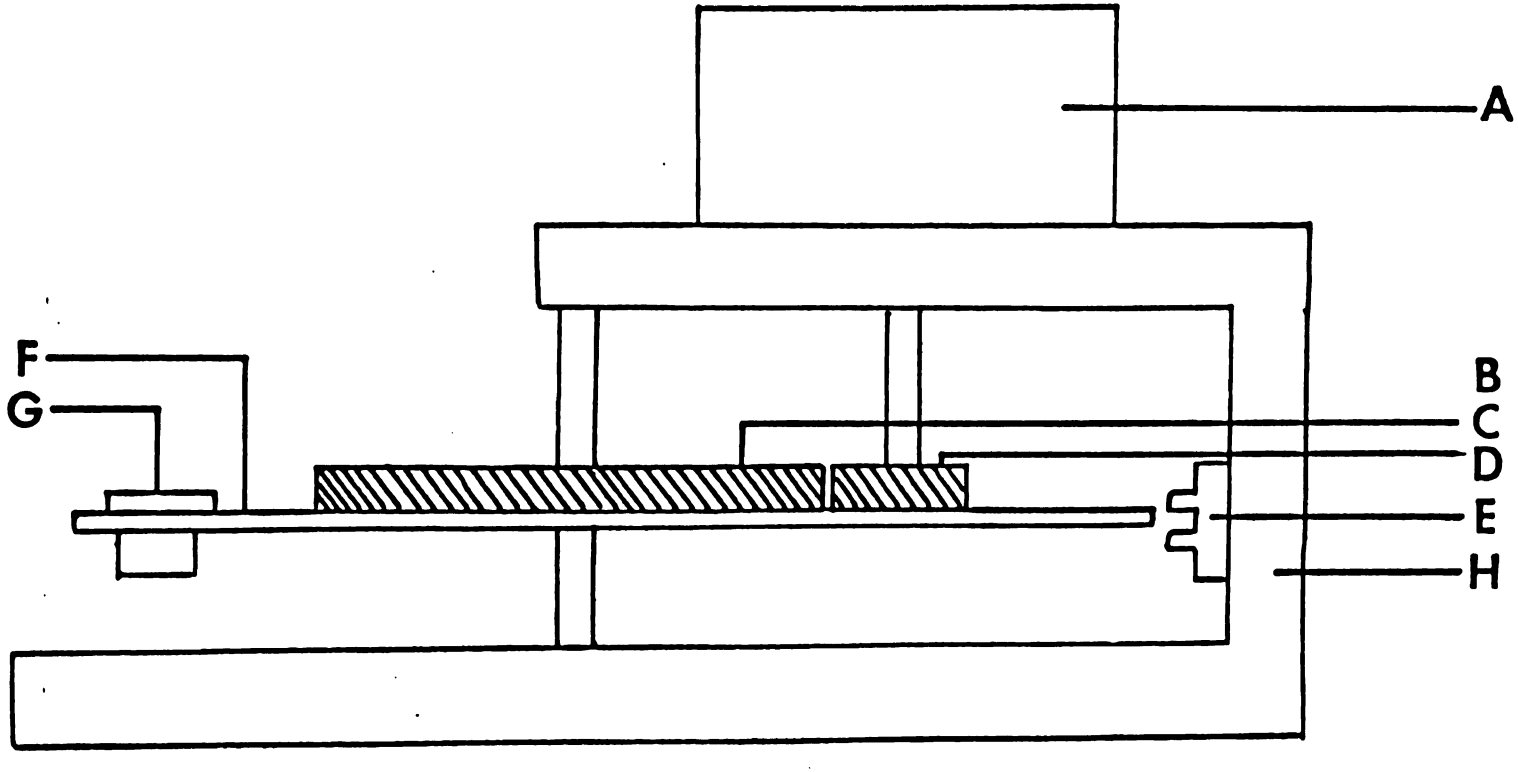


Figure 17. Sample Turntable Module

(A) stepper motor, (B) turntable axle, (C) cup zero interruptor, (D) ½" gear, (E) 2 ½" gear, (F) sample turntable, (G) one of eight removable sample cups, (H) aluminum superstructure

glass barrel G via a ground glass luer joint fitting. The glass tubing I is epoxied to the teflon tubing J so that the delivery needle is physically rigid. Microswitches B detect the limits of the fill or deliver operations when the outer rim of the syringe sleeve D actuates them.

A second teflon delivery needle system with a .5 mm i.d. was constructed from HPLC components (Anspec, Ann Arbor, MI). Also tested in this work was a stainless steel needle (.5 mm i.d.) which had a luer joint fitting and was easily interchangeable with the teflon needles. A second stainless steel delivery system was constructed by epoxying the final 1/2 inch of a 10 µl syringe (Hamilton Co., Reno, NV) to the end of a stainless steel needle identical to the needle described above.

The assembly which transports the syringe between the delivery position (above the atomizer) and the sample loading position (at the sample turntable - Figure 17) is shown in Figure 16. The ASD is secured to the optical rail by an aluminum base plate M and two angle brackets fastened to the base plate. Also fastened to the base plate are two aluminum struts L which are channeled so that they form a track in which the transport base plate F can travel. The 1/2 inch gear J is fastened to the shaft of the stepper motor O which is mounted on, but behind, the transport base plate F. This gear drives

against the larger member of a double gear system I whose smaller member rides against the geared rack L. This geared rack L is fastened to one of the supporting aluminum struts so that the rotation of the motor shaft is transduced into a linear vertical motion with a five to one mechanical advantage. These aluminum struts are 16 inches long and spaced 3 inches apart. The arrival of the transport base plate F at the top of its travel, the level of the sample turntable, or the atomizer is signaled when the appropriate optical interrupter J is activated by the interrupter arm G.

When the delivery head is at the top of its travel it can be rotated between the delivery position (above the sample atomizer) and the load position (above the sample turntable) by pulsing stepper motor N (see top view - Figure 16). Gear R on the stepper motor shaft intermeshes with gear C and drives the delivery head between the two positions which are defined by optical interrupters P and interrupter occluders Q. Also shown in the top view in Figure 16 is an alternate view of the channeled struts and the gear system that drives the vertical movements.

The sample turntable operation is quite straightforward (Figure 17). An aluminum superstructure supports both the stepper motor A and the axle on which the sample turntable is fixed. Rotation of the stepper motor shaft is coupled to the rotation of the sample turntable by a

1/2 inch gear D and a 2-1/2 inch gear E. Cup position zero is defined when a small aluminum tab (not shown) attached to the sample turntable breaks the beam of the optical interrupter F. Each of the eight sample turntable positions may contain a five ml sample cup. In normal operation cup zero is used to store wastes and cup one contains water for rinsing the syringe.

All of the gears, axles, ball bearings, and the geared rack were purchased from PIC Design (Ridgefield, CT).

2. Electronic Hardware

The ASD is controlled by a PDP 8/e computer which has 16K of random access memory, a real time clock, and an extended arithmetic element. Peripherals include a DEC RK05 cartridge disk, a Sykes 7000 dual floppy disc, an ADDS Consul 980 graphics terminal, an LA-30 decwriter and a Heath EU 801 Computer Interface Buffer. This computer system also controls an EAAA instrument (81) in addition to the ASD. The ASD interface was constructed on a general purpose wire wrap board using standard LSI and MSI integrated circuits and communicated to the computer through the Heath interface buffer.

A block diagram of the electronic hardware specifically involved in controlling the ASD is shown in Figure 18. In a typical sequence the computer latches out a command into the interface, the interface decodes the

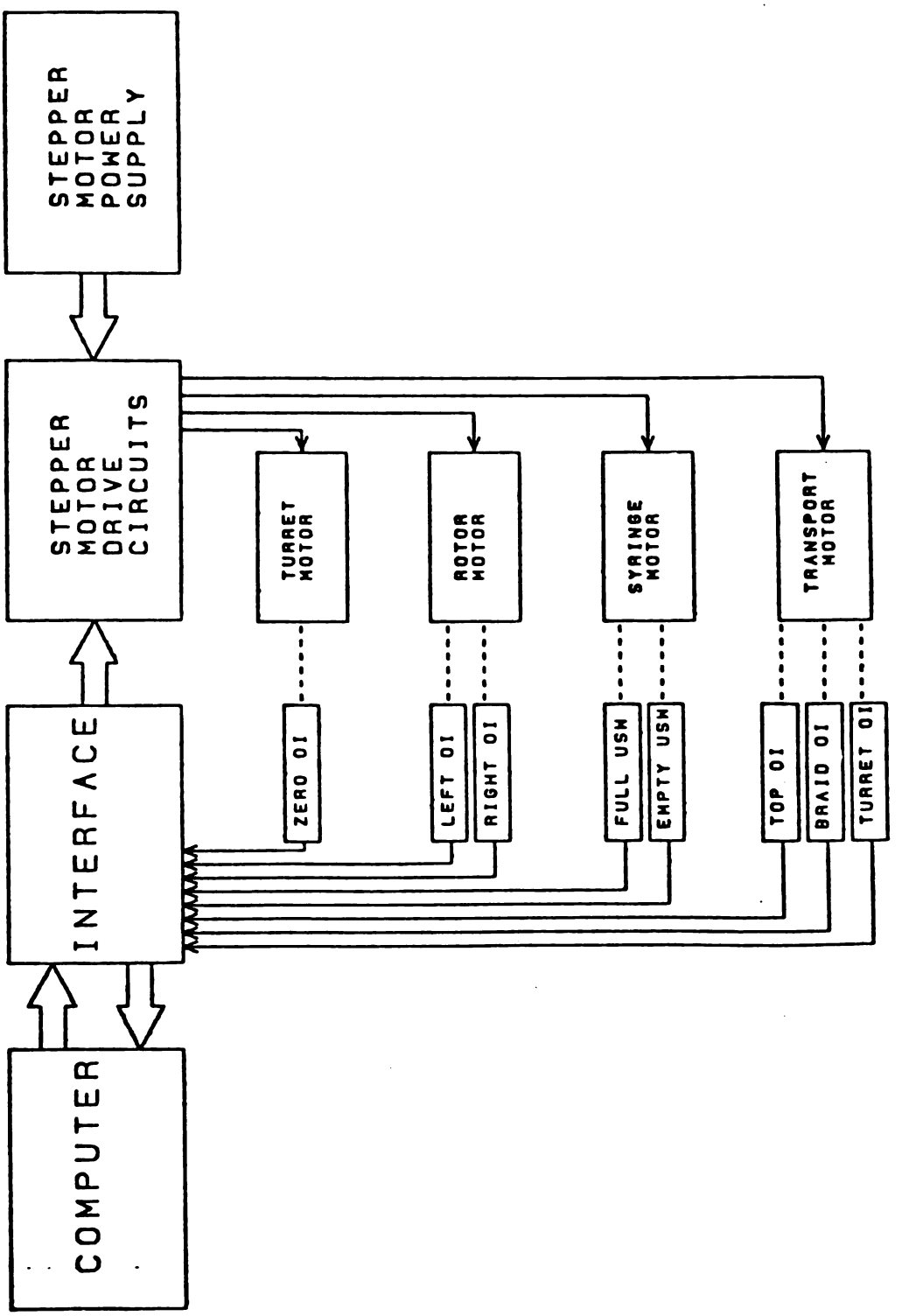


Figure 18. Block Diagram of the ASD's Electronic Hardware

command and pulses the appropriate stepper motor drive circuit. This drive circuit channels power from the stepper motor power supply down the stepper motor power lines in the proper sequence causing the motor to step. The motor is stepped in this manner until the desired movement is completed. The interface recognizes a task as complete when the motor movement causes a change of state at the appropriate optical interrupter (OI) or microswitch (μSW) (this type of interaction is denoted by a dashed line in Figure 17). The syringe motor may also be stopped when a specified number of steps have been executed. The interface also contains error checking logic so that if the execution of a requested command would cause a movement that is self-damaging, then that movement can not be executed. An example of this type of command would be a request to rotate the sample turntable when the syringe needle was in one of the sample cups. Either the completion of a legal task or the request of an illegal movement causes the interface to signal the computer that the task has been acted on. The computer may then interrogate a status register which contains flags indicating the legality of the last requested movement and the status of each of the position indicating optical interrupters and microswitches.

The syringe and turret stepper motors are models 202215D200-F1.6 and 10-2013D40-F75, respectively, Sigma

Instruments, Braintree, MA. The rotor and transport stepper motors are models 23D6102A and 23D6306A, respectively, Computer Devices, Santa Fe, CA. The stepper motor drive circuits were constructed as recommended by the Sigma Stepper Motor Handbook (82). A filtered, but regulated +25V power supply was built in-house to provide the raw power for the stepper motors.

3. Software

Three Fortran IV callable assembly language subroutines control all the actions of the ASD. The first, INIT, safely moves the ASD from whatever position it may be at power up to the state in which the syringe is empty and positioned above the sample turntable sample cup zero. The second, TUNE, moves the sampler so that the delivery needle is at atomizer level and then waits for the operator to fine tune the delivery position on the atomizer. When this task has been completed the ASD returns to the initialized position. The third routine, SAMP, is used to request the delivery of a specified volume of sample solution onto the atomizer. SAMP, INIT, and TUNE all contain a list of job words in sequential memory locations, and each of these jobs words corresponds to one or more motor movements. Table 3 gives the bit assignments for both the job word and the status registers. For example, the job word list for INIT contains a job word for each of the

Table 3. Bit Assignments for the Status Register and Job Word Register.

Accumulator Bit	Status Register	Enable Register
0	Illegal Vertical	Interrupt Enable
1	Illegal Rotor	Top/Purge
2	Illegal Syringe	Up/Down
3	Illegal Turret	Vertical Enable
4	Syringe Empty	Left/Right
5	Syringe Full	Rotor Enable
6	Rotor Left	Turret MSB
7	Rotor Right	Turret Bit 2
8	Vertical Top	Turret Bit 3
9	Vertical Bottom	Turret LSB
10	Turntable at Cup 0	Syringe Special Instruction Flag

following movements: ascend to the top (7400), rotate the syringe over the sample turntable (0336), send the sample turntable to cup one (0302), descend to the sample turntable (0400), empty the syringe (0100), and finally ascend to the top position (7400).

The job word list for SAMP is much lengthier and has three separate entry points which correspond to the purge, load, and deliver functions. To decide which functions are necessary, the software keeps track of the sample number in the syringe and the volume of the solution contained in the syringe. The purge function, which involves the rinsing of the syringe twice with distilled water, is executed only when the sample requested is different from the sample presently in the syringe. The load operation, which entails refilling the sample syringe, is executed only if the sample in the syringe is the same as the last one requested but the amount is insufficient, or when the last function executed was a purge operation. If the request involves no sample changing and the amount in the syringe is sufficient, then only the delivery function is executed.

When the proper entry point has been determined, this routine picks up individual job words, sends them out to the interface, and waits for the interface to set the done flag. When the done flag is set, the routine reads the status register and halts if any of the illegal flags

are set. If the command was not illegal, then the movement is assumed complete, and the next job word is fetched and acted on. This process continues until the requested delivery is completed. The lone exception to this type of operation is the special case of syringe instructions. These instructions require an additional information word which indicates the number and the direction (i.e., load or deliver) of syringe motor pulses to be executed.

A request for a sample delivery from a Fortran routine is in the form of a subroutine call with two arguments one which indicates the sample number, the other which indicates the amount to deliver in μℓ. The only restriction is that before the first delivery is requested the INIT and TUNE routines must have been properly performed.

C. EVALUATION OF PERFORMANCE

The important criteria to be considered in the evaluation of the ASD are the delivery accuracy, precision, and inertness toward the sample solution. The accuracy and precision of the delivery and thus the total AA measurement precision is quite dependent on the mode of operation and the particular delivery needle used. The most obvious conclusion to be drawn from the study of precision given in Table 4 is that an entrapped air pocket is quite detrimental, especially for needles with small diameters. This entrapped air pocket is formed in the precision glass

Table 4. Measurement Reproducibilities of Various Delivery Modes.

Needle Material	i.d.	%RSD[1]	TRSD[2]	%RSD[3]	%RSD[4]
St. Steel	.5	4.3	12.4	2.8	120
St. Steel	<.2	2.2	*	2.2	*
Teflon	.5	1.8	100	1.5	65
Teflon	.3	3.4	95	3.3	23

RSD[1] Atomizer not heated during delivery, no entrapped air space in syringe.

RSD[2] Atomizer not heated during delivery, entrapped air in syringe.

RSD[3] Atomizer heated during delivery, no entrapped air space in syringe.

RSD[4] Atomizer heated during delivery and entrapped air space in syringe.

* Deliver totally ineffectual.

barrel between the syringe plunger head and the surface of the sample solution whenever the ASD initially fills itself. The only way to prevent this air bubble is to manually charge the syringe barrel with sample solution before the fill function is executed. When the ASD is operated with an entrapped air pocket the inner diameter of the delivery needle significantly affects the delivery precision. This is explicable if one considers that at the time of delivery the sample solution is subjected to capillary forces of the braided graphite atomizer and the syringe needle. Also the surface tension of the solution and the hydrophilic or hydrophobic nature of the syringe needle material affect the reproducibility of the delivery. Heating the atomizer to desolvation temperature during the delivery enhances the ability of the atomizer to compete for sample solution and when an entrapped air pocket is present this enhancement is very detrimental to delivery precision. In this part of the study no scavenging of the analyte by either the stainless steel or teflon needles was observed.

Table 5 shows the results of long term tests on the precision of sample delivery for three individual atomizers. In Figure 19 the graphical representation of one of these experiments demonstrates that no perceptable scavenging of the 1 ppm cadmium by the syringe needle or precision glass barrel occurred during the duration of this

Table 5. Long Term Stability.

Atomizer No.	1		2		3	
Run Number	Integrated	Abs. RSD	Integrated	ABS. RSD	Integrated	Abs. RSD
1-40	12.09	2.4	12.30	3.4	12.85	5.8
41-80	11.78	2.6	12.16	2.7	12.19	3.8
81-120	11.68	2.7	12.21	1.7	12.80	3.9
121-160	11.79	2.4	12.30	1.9	11.98	2.9
161-200	11.84	2.6	12.41	1.6	12.05	2.4
201-240	12.04	2.9	12.41	1.7	12.07	3.6
241-280	12.15	2.9	12.48	1.7	12.14	1.9
281-320	12.14	2.7	12.60	1.5	12.18	2.7
321-360	12.35	3.3	12.64	1.8	12.30	2.2

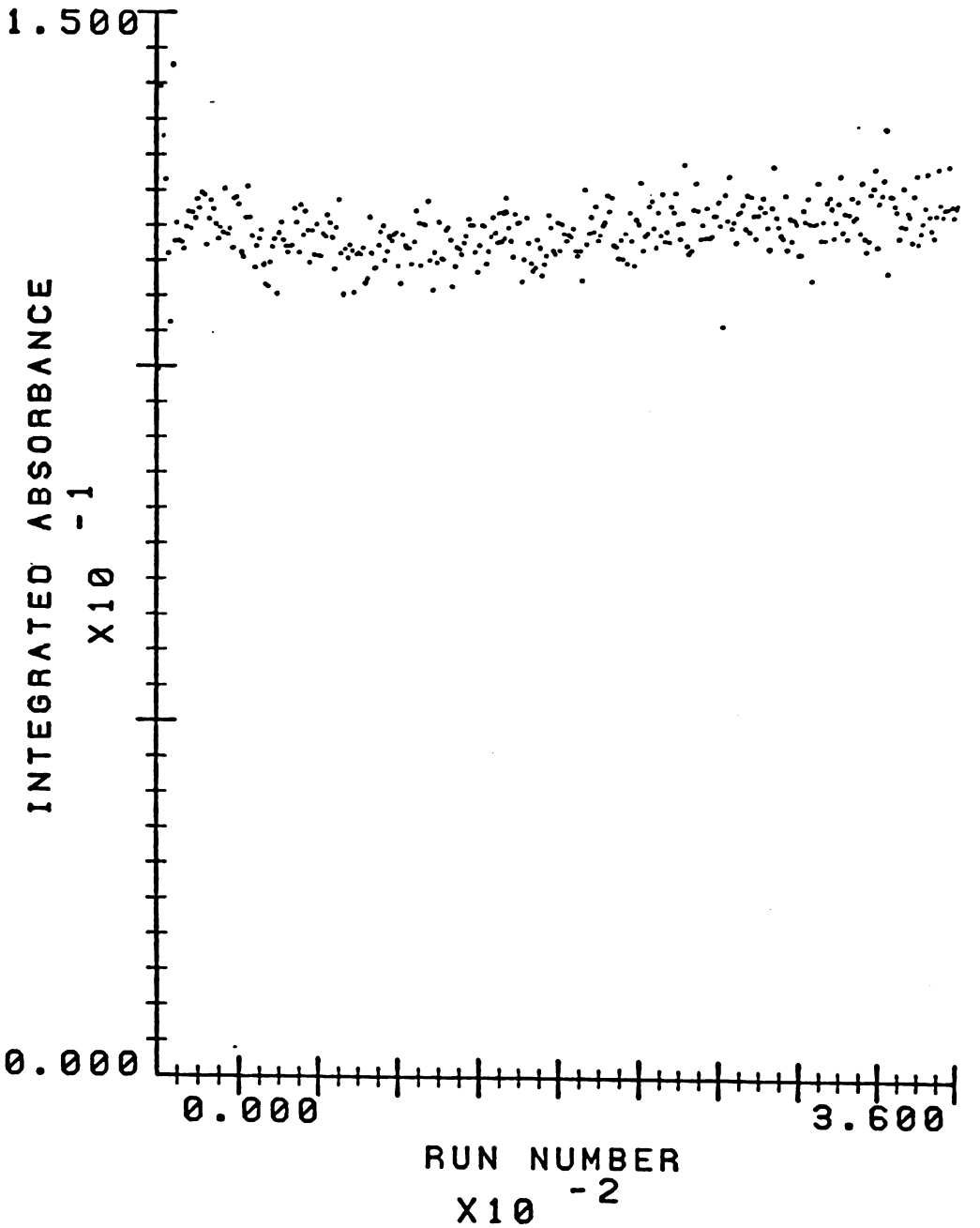


Figure 19. Long Term Stability

experiment (approximately 5 hours). What small drift there is in this data may be attributed to aging of the atomizer which changes the sensitivity of the measurement. These results were acquired using the small diameter stainless steel needle, no entrapped air space, and desolvation heating during delivery. Results with the other delivery needles under similar conditions were quite similar to those shown.

The ASD was also tested for the linearity of volume delivery with the smallest diameter steel needle and 1 ppm Cd. The results for sets of six requested deliveries of volumes between 1.0 and 2.0 µℓ are shown graphically in Figure 20. The data obey the following equation:

$$y = (.60 \pm .02)\ x + (.04 \pm .03)$$

where y is the integrated absorbance in arbitrary units and x is the delivery volume requested.

When automatic sample changes are required it is necessary to operate the ASD with an entrapped air pocket to prevent possible solution mixing effects. In the multiple sample mode the ASD showed no memory effects when it was programmed to rinse twice with distilled water and twice with the new sample solution between samples, however, the precision was degraded to typical values of 10 to 15%.

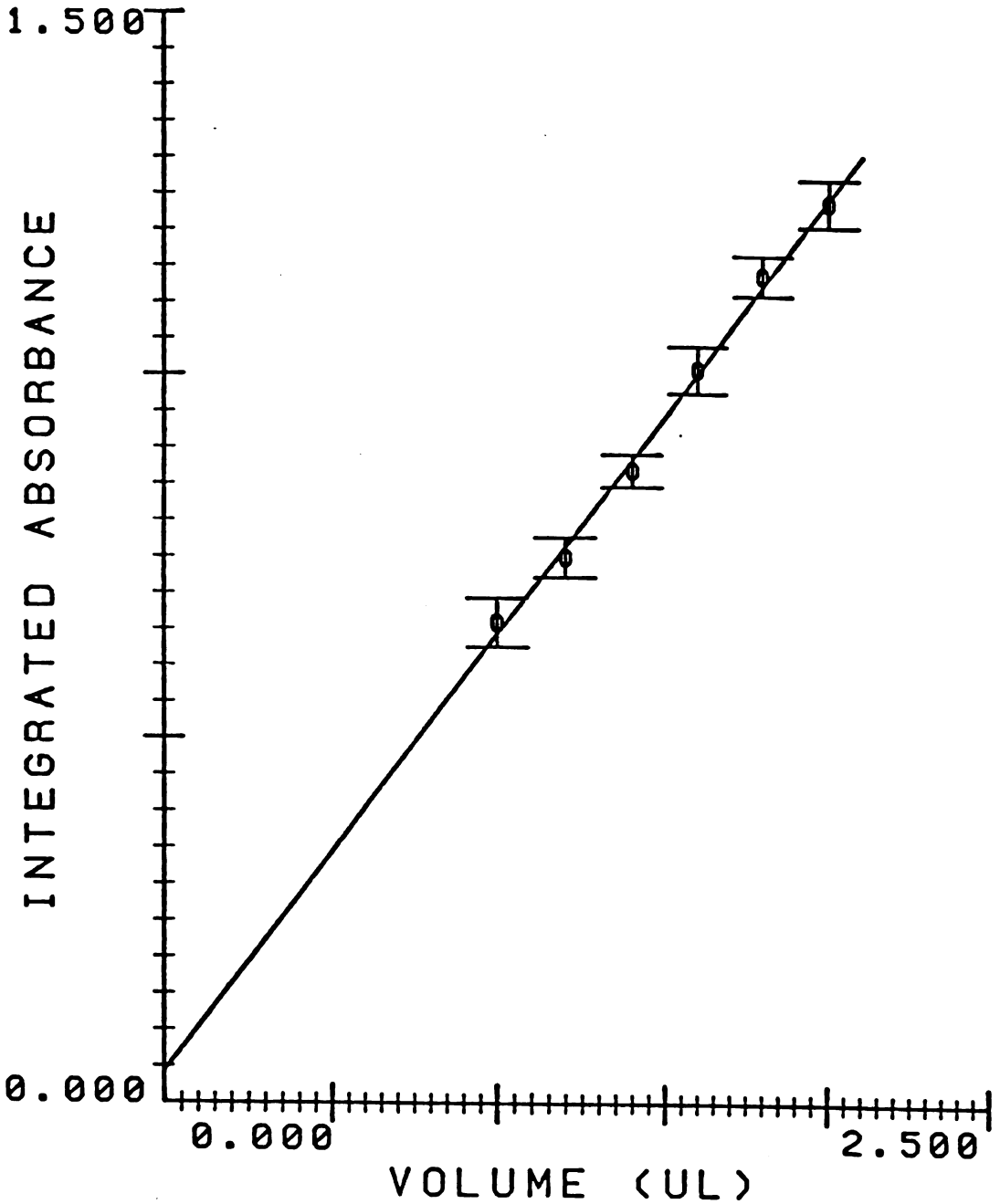


Figure 20. Volume Delivery Linearity

D. CONCLUSIONS

The automatic sample delivery system can precisely and reliably deliver μℓ sized samples onto an electrothermal atomizer, and is suitable for long term computer-controlled experimentation. Although the ASD was only tested for 1 ppm Cd, with proper treatment of the surfaces in contact with the sample solution (1,10), it is expected to perform equally well for solutions of other elements and more dilute solutions of cadmium.

CHAPTER 5

SIMPLEX OPTIMIZATION STUDIES ON THE EAAA INSTRUMENT

The attractiveness of on-line optimization of experimental parameters was pointed out in the introduction to this thesis, and the small amount of work which deals with the application of simplex optimization techniques to chemical problems was reviewed in Chapter 1. The work in this chapter demonstrates the ease with which the simplex optimization technique was adapted to the optimization of experimental parameters in the EAAA instrument and also the value of this approach as a tool to help solve general EAAA analysis problems.

This chapter contains a description of the Nelder and Mead simplex paradigm and the software used to implement this optimization technique on the EAAA instrument. The cadmium integrated absorbance as a function of horizontal and vertical displacements of the observation window away from the atomizer was recorded and the on-line simplex optimization technique was tested and characterized on this known response surface. The cadmium results were compared to results of optimizations of the manganese

integrated absorbance in similar experiments where the shape of the response surface was not totally characterized. A four parameter, on-line optimization of the cadmium integrated absorbance as a function of the sample amount, time delay between sample runs, and the horizontal and vertical displacements of the observation window away from the atomizer, was also carried out. An on-line optimization with a complex response function, which was to minimize analysis time and maximize the integrated absorbance signal of cadmium, was also attempted. An attempted optimization of the measurement precision in a semi-automated, multicomputer system is also described. The optimization of the time limits of the integration of the cadmium signal was carried out off-line on pre-recorded data. Finally, some general observations about on-line simplex optimizations and some suggestions for improving the performance of the simplex algorithm in the presence of noise are given.

A. INTRODUCTION

A brief introduction to simplex nomenclature and the simplex algorithm is necessary before a detailed discussion of the experimental work may be attempted. A system is any process or device that operates on an input or inputs to give an output or outputs. The response is a system output, or combination of system outputs, that

is chosen as the entity to be optimized. Factors are a selected subset of the system inputs chosen by the experimenter to be varied in order to maximize the chosen response.

1. Example

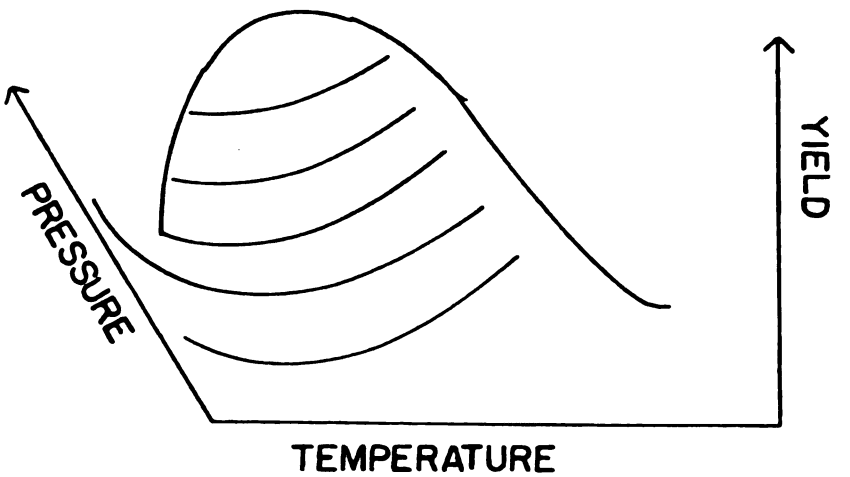
In order to define factor, system, and response in a more familiar environment, a simple chemical reaction $A + B \rightarrow C$ is used here as an example. There are several experimental variables which, if they affect the chosen system response, may be termed factors. Possible factors in this system include: the mole fraction of A, the mole fraction of B, the temperature, the pressure, the reaction time, the presence or absence of catalysts, inhibitors, light, etc.. The particular system output chosen as the response is entirely at the experimenter's discretion. Some reasonable choices for this example include the following: percent yield, time of reaction, amount of useful by-products, input energy costs, ease of cleanup, etc.. In real chemical systems the response chosen is often a weighted combination of several of these responses.

A plot of the response value as a function of the values of the selected factors yields a response surface. The response surface is conveniently described as the system response output under every conceivable combination of input values of all the chosen factors, when all other

variables are held constant. In our example, if we chose temperature and pressure as factors, and the percent yield as the response, then the response surface might be shaped like the response surface represented in Figure 21. Figure 21a is a pseudo-three-dimensional plot of the response surface. In it the optimum yield is located at the "top of the hill". This same response surface is also shown in a two-dimensional contour plot in Figure 21b. In each plot iso-yield lines connect the combinations of temperature and pressure that give identical yields. Note that the "top of the hill" in Figure 21a is located inside the smallest of the ellipses in Figure 21b. The simplex algorithm is an optimization technique that uses a set of rules to move a geometric figure located on the response surface toward the response optimum. The geometric figure is called a simplex and has n+1 vertices where n is the number of factors.

2. Comparison of the Simplex Method with Other Optimization Techniques

In order to establish the value of the simplex optimization technique, it should be compared to the other commonly used optimization methods. The most widely used optimization technique is the iterative method. In this method, all variables except one are held constant,



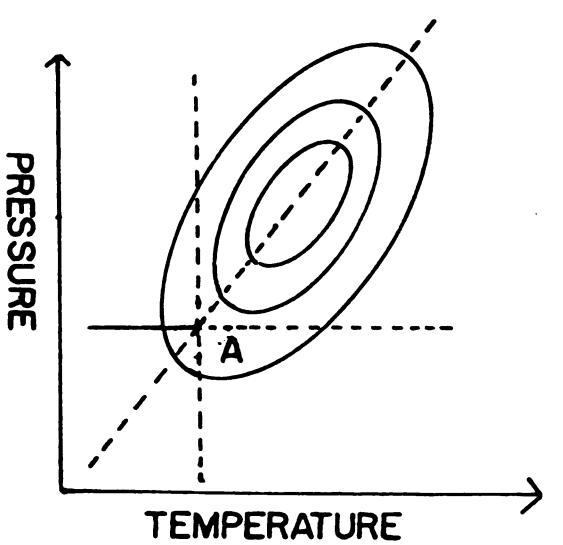


Figure 21. A Typical Response Surface

and the chosen variable is adjusted until the response optimum is found. Then, a second variable is chosen, and its value adjusted until the response is again optimized. This process is then iterated until a change in the value of any of the variables results in a decrease in the system response. The failure of this technique to find the true optimum is demonstrated by the example response surface shown in Figure 21b. When point A is reached, the adjustment of either the temperature or pressure yields a decrease in the yield. To visualize the failure of the iterative optimization method to find the true optimum imagine a mountain climber who must climb under the following restrictions. First, every move must result in a net gain in altitude, and second, he may move only directly north, south, east, or west. The particular mountain that this climber must ascend has a ridge running in a northwesterly direction from the base to the peak. When the climber finds himself on this ridge, movement south or east will be downhill and movement north or west will be down the side of the ridge and will result in a net loss in altitude. Under these restrictions, which are the same as those imposed by the iterative optimization technique, the climber will never reach the mountain top. This type of ridge is known to occur frequently in chemical systems (47).

To find the true maximum yield, in the chemical

example, both parameters must be adjusted at the same time. Since the simplex has three vertices in a two dimensional optimization it can change the direction of its movement to any of the points of the compass and so it can move diagonally toward the true optimum.

The most rigorous optimization techniques involve a patterned testing of the entire response surface. This is impractical for complicated systems that have more than two or three factors since these techniques require an inordinately large amount of experimental data to be collected.

3. The Nelder and Mead Simplex Algorithm

The moves of the Nelder and Mead simplex algorithm (36), which was used exclusively in this work, can be explained with the aid of Figure 22. For two factors the simplex has three initial vertices, which are tested and ranked according to the response at each vertex. The initial vertex is the triangle BNW, where B is the best vertex, N the next best, and W the worst vertex. Vertex W will be discarded by the simplex algorithm, and a new vertex for the simplex will be generated somewhere along the line defined by the old vertex W and the centroid of the opposite face of the simplex. The first test is of the response at the point R, which is an equal distance from the centroid as point W but on the opposite side of

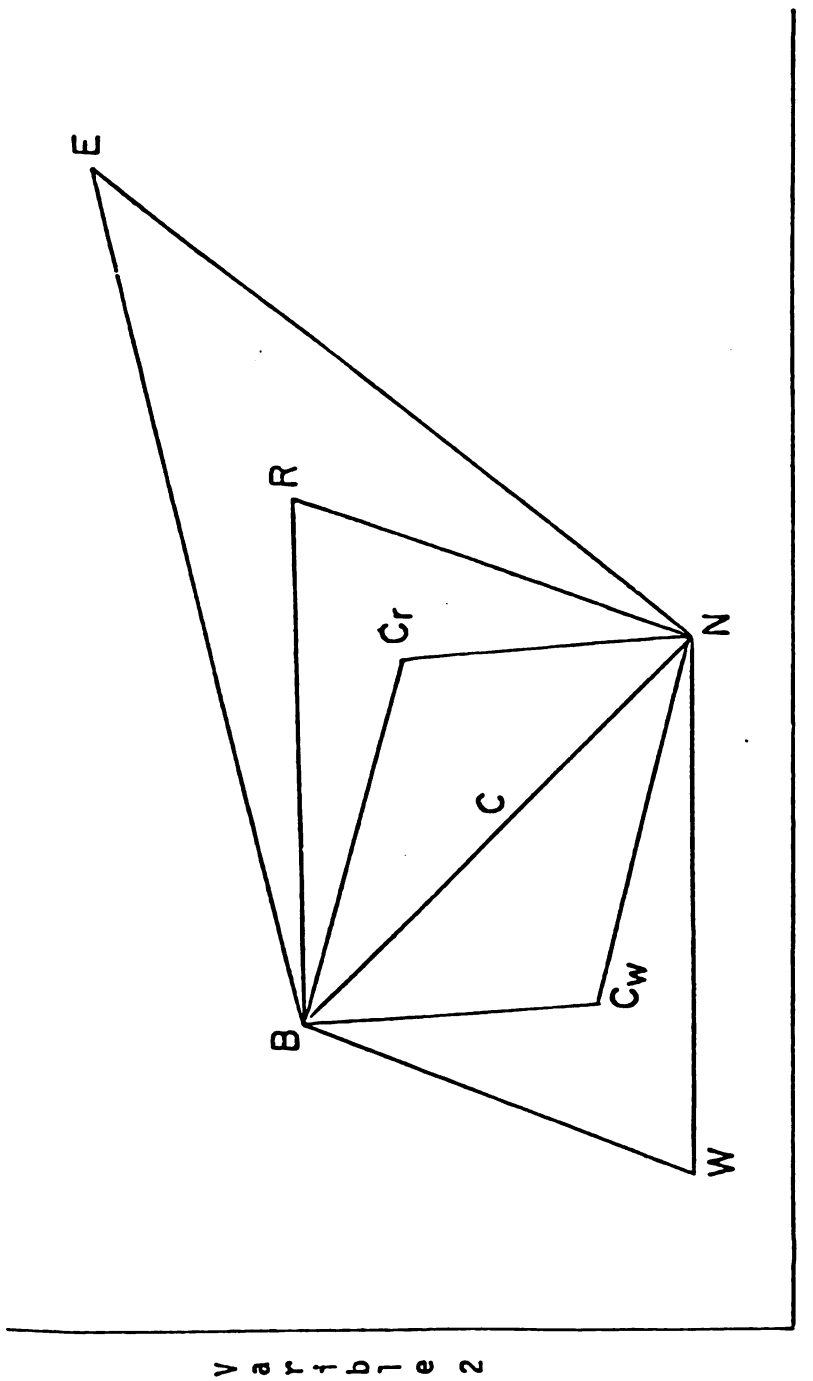


Figure 22. The Posible Moves of the Modified Simplex Algorithm

the line BN. If the response at R is greater than the response at B, then an expansion to point E, which is twice the distance from the centroid as vertex R, is attempted. If this response is better than R, then the new simplex will be the triangle NBE.

If the response at R is between the responses of B and N, but greater than the response at E, the new simplex will be RBN. If the response at R is less than at N but greater than at W, a contraction to the BNCr simplex is executed. If the response at R is worse than the responses at N and W then the response at point Cw will be evaluated and the new simplex will be BNCw. The last two types of moves (contractions) are executed only if the response at this contraction is better than at W.

In the original Nelder and Mean algorithm, if all of these attempted moves failed to achieve a better response than W, a more massive contraction is executed. In this work, the next worst vertex N of the original simplex is reflected throughout the centroid of the simplex face opposite it. A second modification of the Nelder and Mead algorithm was necessary to deal with instrumental errors. If a vertex had been retained by the simplex for more than n+1 moves, it was re-evaluated and the old response was replaced by the new response. In the EAAA instrument occasionally a drop of solution would cling to the exterior of the delivery needle during the refill operation

and would be delivered to the atomizer along with the requested sample, and this would result in an erroneously high evaluation of the response at that point. If this response was not re-evaluated, the simplex would rotate about this point in a manner not unlike a dog chasing its tail. Note that most researchers have averaged the old and new values of the response to get the response value to be used in the simplex algorithm (47,52,59). This action is best suited for averaging out random error; the replacement operation is best suited for reducing systematic instrumental errors, since with averaging it might take an unreasonable amount of testing to overcome the effects of the one incorrect response evaluation.

B. SIMPLEX SOFTWARE

The fundamental FORTRAN IV routine used to implement the simplex algorithm on the EAAA instrument was adapted from a routine written by Mr. Martin Joseph and Dr. Eric Johnson in our laboratories (85). Some extensive modifications were necessary to allow this routine to be embedded into the EAAA routines and to allow the desired flexibility in the selection of the variables to be optimized and the calculation of the response.

1. Initiation of a Simplex Optimization

The user may enter the Simplex mode of operation by specifying the :SMP command, to the GBA monitor (subroutine INPUT). Upon receiving this command the program requests the user to select the number and the names of the variables to be optimized. At this point the user may chose 2 to 4 of the 13 possible experimental parameters listed in Table 6 as factors for the simplex optimization. The selection of these variables must be influenced by certain practical considerations for the simplex to operate properly. The software, however, will accept any combination of these variables. The rules for making acceptable choices of factors in this particular application are discussed later in this chapter. The user is then prompted to enter the upper limit, lower limit, and the desired convergence precision for each variable. For example, the lower limit for the variable DEST (desolvation time) might be set at 0. s, since negative times are meaningless. The upper limit might be set at 60. s to prevent the simplex algorithm from making a response test at a desolvation time that is so long that it is experimentally impractical. The value entered for the desired precision relates to the detection of the convergence of the simplex on the optimum. When all the vertices of the simplex have values for each factor which are separated by less than the specified precision

Table 6. Experimental Parameters that May be Optimized With the Simplex Routine.

Variable	Description
DEST	Desolvation time
ASHT	Ashing time
ATMT	Atomization time
DELT	Delay time between sample runs
DESP	Desolvation power
ASHP	Ashing power
ATMP	Atomization power
AMNT	Amount of sample delivered
HPOS	Horizontal location of observation window
VPOS	Vertical location of observation window
FREQ	Data acquisition rate
INTST	Number of the data point at which the integration is begun
INTSP	Number of the data point at which the integration is stopped

for that factor, the simplex is halted. The user is then requested to enter the factor values for each of the initial simplex vertices. There is also a choice of the optimization mode, and the options include: limited integration mode (this mode must be used if INTST and INTSP are chosen as factors to be optimized), whole integration mode (normal mode), or peak absorbance. The user may also specify more than one sample run to be taken per response evaluation. Finally, one can enter an iteration limit, which will cause the simplex to halt after this specified number of moves even if it has not converged on an optimum. This last option is quite useful if it is desired to perform several optimizations under batch stream control as described in Chapter 3.

2. Run-Time Options

Some of the option variables function in the simplex mode of operation similar to the manner they function in the normal mode. If the LPT option is set, an output of the initial simplex parameters, as well as a running account of the progress of the simplex, is output to the lineprinter. The PLT option causes an absorbance versus time plot to be output to the system terminal. This has the same appearance as the plot given in the normal mode, but uses a separate set of subroutines because of overlaying requirements. Switch 0 on the front panel of the general

interference box still serves the function of aborting the task and exiting to the GBA monitor. Switch 4 causes an absorbance versus time plot even if PLT was not requested. The file option variables in the simplex mode pertain to file output formats entirely different from those in the normal mode. One file format outputs the iteration number, the factor values, and the response at the vertex just tested. This file is useful for the creation of plots of the movement of the simplex and plots of the progress of each individual variable. The other file output keeps a record of all of the response values and is useful for archival storage since it is short.

C. TESTS OF THE SIMPLEX ALGORITHM ON A KNOWN RESPONSE SURFACE

On-line optimizations of experimental parameters for a particular instrument must deal with noisy response surfaces and, if parameters other than those parameters being optimized change, the algorithm must also deal with shifting response surfaces. The success of simplex optimizations is dependent on the characteristics of the system being optimized. Thus it is desirable to test the simplex on a "standard" response surface, where the global optimum is known.

1. The Cadmium Integrated Absorbance Response Surface

Two-dimensional mapping experiments of cadmium absorbances (83) yielded a known response surface for one particular optimization that might be attempted on the EAAA instrument. The optimum integrated absorbance in the response surface was assigned an arbitrary value of 100, and the iso-absorbance lines were drawn to define the regions where the integrated absorbance decreased by 10% increments from this optimum value.

Figure 23 shows the response surface and several starting simplexes used in the optimization of the cadmium integrated absorbance signal. The global optimum is located near 0. mm vertical displacement and 0. mm horizontal displacement. The response drops off rapidly in all directions (the slope of the response surface is inversely proportional to the distance between the iso-absorbance lines). The slope for vertical displacements is less severe than the slope for horizontal displacements. Also the pointed intersection between the iso-absorbance lines near zero horizontal displacement are indicative of a sharp ridge in the response surface. The irregularities in the iso-absorbance lines above 12 mm vertical displacement suggest a non-smooth surface. Also, since source flicker noise (hollow cathode lamp flicker) is a significant source of imprecision, the signal-to-noise ratio should be poorer for the regions bounded by iso-absorbance

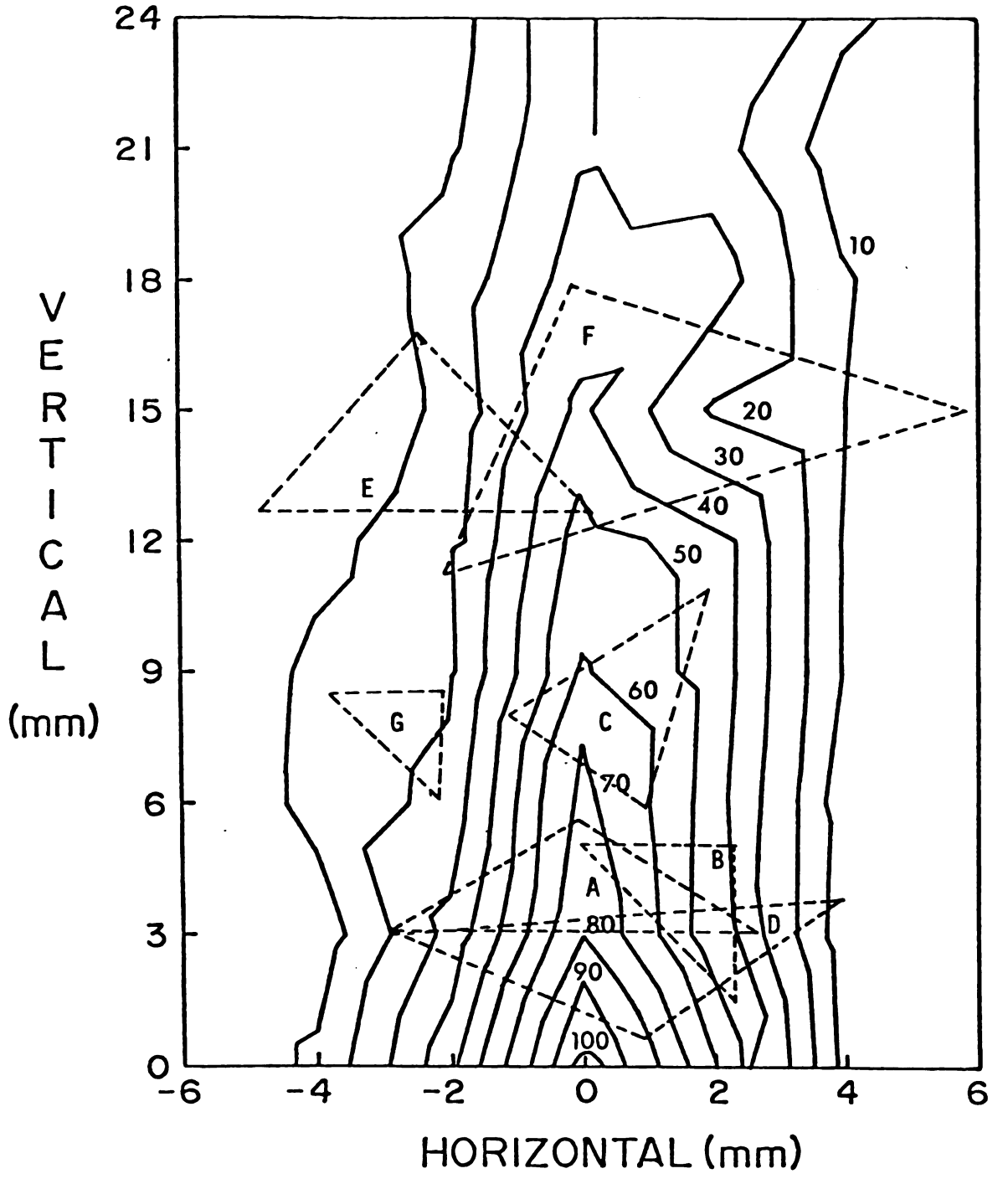


Figure 23. Cadmium Integrated Absorbance Response Surface with Assorted Starting Simplexes.

lines which indicate low cadmium concentrations. An additional source of noise in the upper regions of the vapor cell is caused by the breakup of flow laminarity as the sheath gas begins to encounter the surrounding atmosphere near the top of the atomization cell. The slight asymmetry of the surface in the upper regions is probably indicative of the prevailing wind currents in the laboratory.

2. Methods of Monitoring the Progress of the Simplex

The movements of the simplex which used triangle G in Figure 23 as the starting simplex are shown in Figure 24. Note that it initially moves rapidly toward the horizontal center of the vapor cell and down toward the atomizer with two expansions. Then two contractions on the only vertex remaining from the original simplex cause a rapid approach toward the horizontal optimum. After one more expansion in the general direction of the global optimum, the algorithm causes a series of reflections and contractions until an optimum very near the true global optimum is found. This type of plot is instructive, but somewhat confusing, and applicable only to two-factor optimizations. Single parameter progress plots such as Figures 25, 26a, and 26b, are more easily analyzed. These plots show both the successful and nonsuccessful moves of the algorithm for a single factor at a time, and thus may be used to

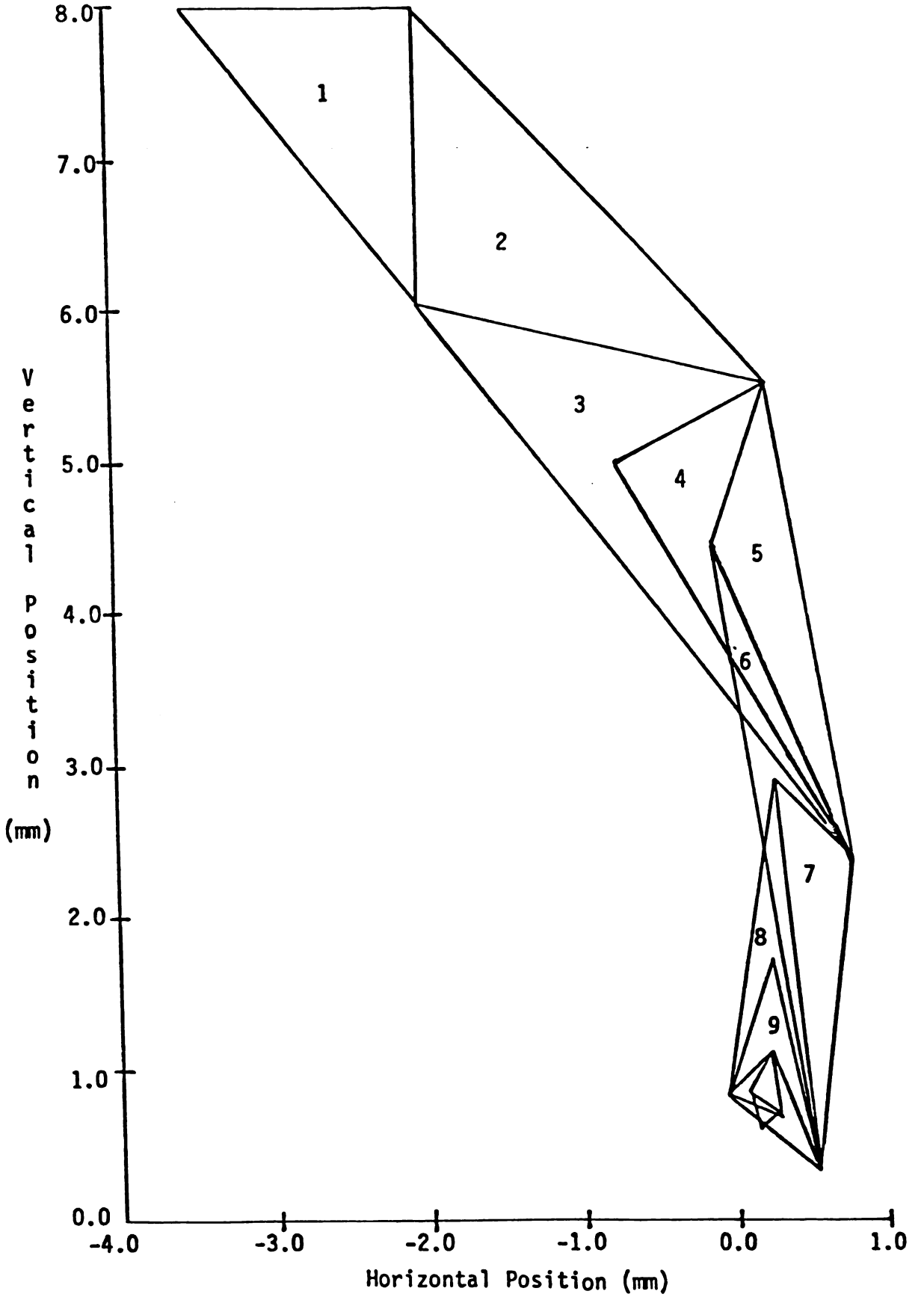


Figure 24. The Moves of Simplex Optimization G.

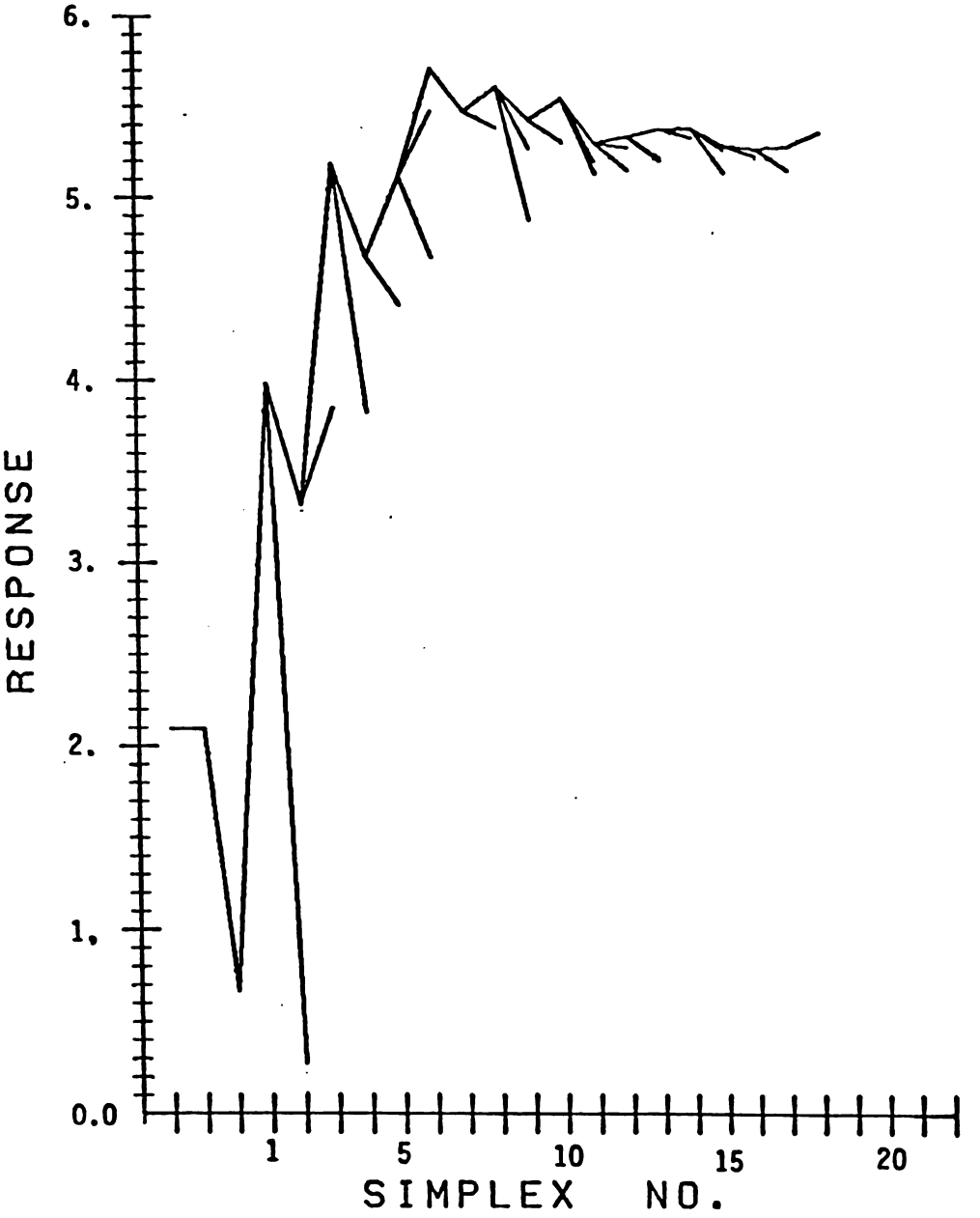


Figure 25. Progress Plot of the Cadmium Integrated Absorbance Response in a Two Parameter Optimization

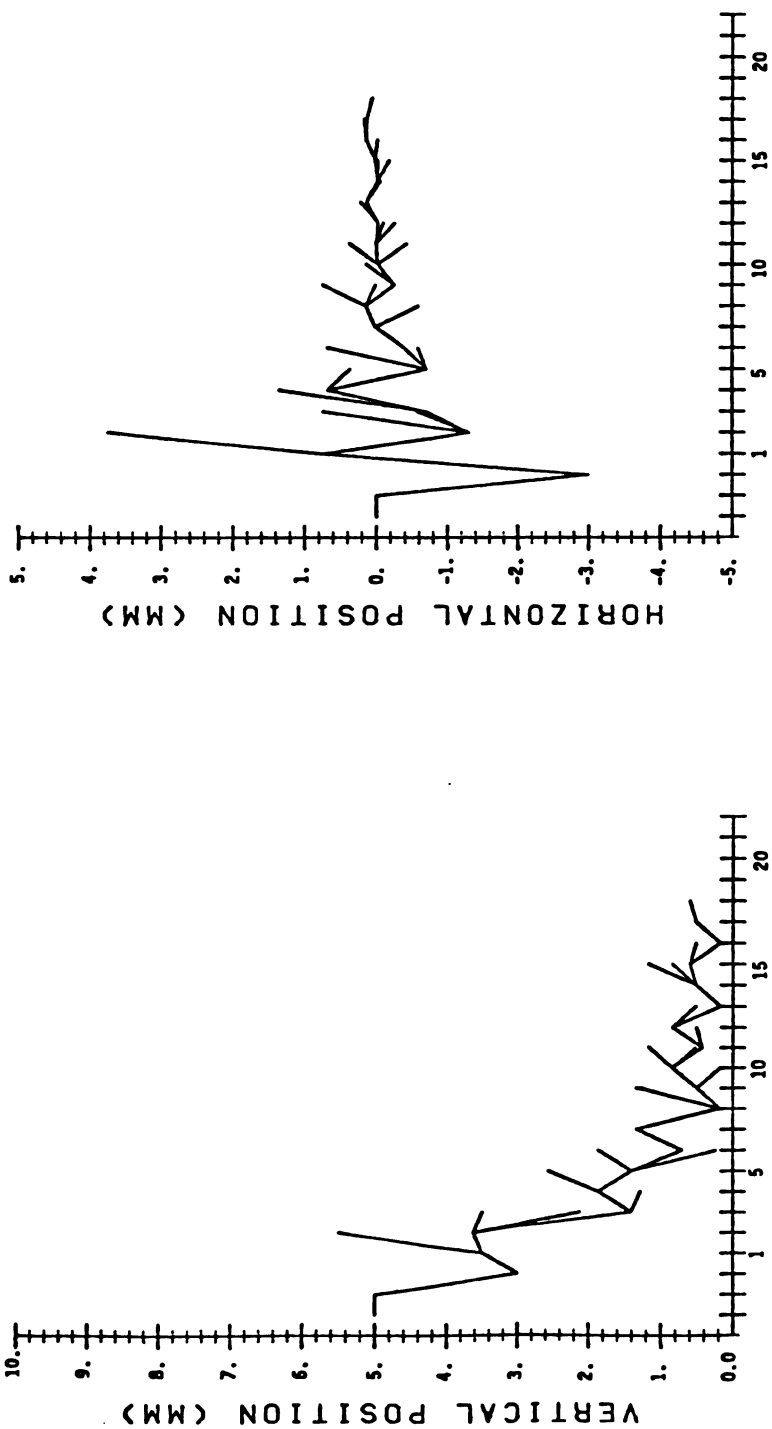


Figure 26. Progress Plots of Vertical Position and Horizontal Position in a Two Parameter Optimization of Cadmium Integrated Absorbance

monitor the progress of an optimization of any dimensionality. These plots were created by slightly modifying the optional simplex mode output file so that it could be used as an input file to the program MULPLT on the CEMCOMGRAF facility. The progress curves are generated by first plotting the value of the parameter of interest at each vertex of the starting simplex and then connecting the points with lines. A line is then drawn from the last plotted vertex value to the value of that parameter at each attempted move. The line drawing process is then repeated using the value of the parameter at the last successful simplex move as the new origin for further lines to values at the next attempted move, etc..

Figures 25, 26a, and 26b indicate the movements of simplex optimization A in Figure 23. Some interesting conclusions about the optimization process may be deduced from a careful consideration of these figures. The movement of the response (integrated absorbance) is shown in Figure 25. For the initial moves, the improvement in the response is rapid, because the simplex is large and located on a steep gradient of the response surface. However, when the simplex is smaller and on a more level portion of the response surface, the changes in the response are quite small. It seems quite incongruous that the response seems to degrade from its peak value at simplex 7 to a somewhat smaller value when the simplex is halted.

This could be due to several factors. It is possible that this vertex was actually retained in the final simplex and only the other two vertices moved. However, inspection of the data in the data file that recorded the simplex moves revealed that this had not occurred. It is also possible that there might have been a change in some parameter not being tested. Atomizer aging, which can cause a loss in measurement sensitivity, could cause a shift in the response surface and this process might yield a progress plot like the one obtained. Indeed this may be part of the cause of the degradation in response. But, as will be shown in the analysis of the vertical parameter progress plot, this type of sawtooth plot is also indicative of the approach of the simplex to an optimum that lies on one of the parameter bounderies.

If the progress of the vertical variable, Figure 26a, and the progress of the horizontal variable, Figure 26b are compared, it can be seen that the simplex converged on the horizontal optimum much faster than it did on the vertical optimum. This cannot be attributed solely to the fact that the response surface has a larger slope in the horizontal variable than in the vertical variable. First, convergence for a single parameter is dependent on the size and orientation of the initial simplex. Second, because the size of the simplex moves are limited and any attempted moves outside the designated boundaries are

assigned very unfavorable responses, the simplex technique has difficulty converging on an optimum that is located on a boundary.

3. Comparison of Optimizations with Different Starting Simplexes

The results of starting the optimization at each of the starting simplexes shown in Figure 23 are given in Table 7. Optimizations A1 through A6 all shared the common starting simplex A. On a noise free surface the algorithm would make exactly the same movements each trial, and the optima found would be identical. Experimental noise causes six different optima to be located, but all of these are in the immediate vicinity of the true global optimum. Although the least efficient optimization (A5) took nearly 60% more moves than did the most efficient (A4), both processes took considerably fewer experimental observations than any mapping experiment that could be expected to find an optimum as near the true global optimum.

The D, E, and F initial simplexes all failed to converge on the true global optimum. For optimizations E and F, the reason is that the initial simplex was located in a region of the response surface that has an unfavorable signal-to-noise ratio, and this caused the simplex

Table 7. Results of Optimizations of the Horizontal and Vertical Positions for Maximum Cadmium Integrated Absorbance.

Optimization	HPOS Optimum (mm)	VPOS Optimum (mm)	Iterations Required
A1	- .29	.21	14
A2	.09	.49	14
A3	.19	.28	14
A4	.0	.27	13
A5	- .13	.54	23
A6	.13	.27	18
B	.03	.67	16
C	.22	.67	16
D	- .19	1.4	8*
E	-1.8	14.0	8*
F	- .80	13.5	12*
G	.01	.54	11
A6'	.20	.21	25
G'	.14	.29	14

*Failed optimization due to noise on the response surface.

to contract prematurely. Noise causes more contractions than expansions, since contractions are statistically favored by the simplex algorithm. It is possible, but highly unlikely, that there are indeed some small local optima which the simplex found in this region. These optima would have to be quite small or they would have been visible in Figure 23. The reasons for the failure of optimization D to converge on the true global optimum are more subtle. First, the initial simplex was created so that its size, location, and orientation made it difficult for the simplex algorithm to move it toward the true optimum. Because it was very close to the vertical boundary and oriented so that its first moves were contractions, the simplex fell prey to instrumental noise and contracted before it could re-orient its movement in the direction of the optimum and make significant progress. In fact, all the eight moves of this optimization were contractions.

Optimizations B, C, and G all converged in the vicinity of the true global optimum in a reasonable number of movements. Along with optimizations A1 through A6, these optimizations demonstrate that a judicial choice of a starting simplex involves the placement of the simplex in a region of the response surface that has a significant slope, low noise, and is not near any variable boundaries.

Another form of failure for the simplex, which was

not observed in these tests, is the loss of dimensionality of the simplex on a response surface ridge. If the simplex were to somehow locate two of its vertices on the response ridge (i.e., both vertices have a horizontal value near 0. mm), the algorithm would cause a series of reflections and contractions on the one vertex not on the ridge until the simplex resembled a straight line. When this happens, the movement of the simplex is restricted to movements in the direction defined by this line, and the true optimum may never be located. None of the optimizations in this study showed this failure mode, because although there is a ridge in the response surface, there is still a consistent slope in the variable which parallels the ridge. (In this case the change in response for vertical displacements along the vertical ridge is consistent and large enough to prevent the loss of dimensionality.)

The software caused the simplex to be halted when the range of values for each parameter at all the vertices was within a specified precision. In these optimizations the precision specified for both the horizontal and vertical variable was 0.5 mm. If this restriction is lifted, the optimum located is closer to the true global optimum. Optimizations A6' and G', which are continuations of optimizations A6 and G, demonstrate this point. In these optimizations, the simplex was allowed to continue until

the operator could no longer detect any significant movement of the simplex. The optima located are significantly closer to the true global optimum, and the range of values for the horizontal factor under these conditions reaches 0.03 mm and 0.32 mm for optimizations A6' and G', respectively. Similarly, the range of values for the vertical factor is 0.13 mm and 0.14 mm.

4. Optimization of the Integrated Absorbance of Manganese

The results of studies on the optimization of the horizontal and vertical position for maximum Mn integrated absorbance are shown in Table 8. The vertical concentration profiles for manganese (84) and cadmium (28) show that the atomic population of manganese decreases with the increasing distance above the filament more rapidly than the atomic population of cadmium. For this reason, the initial simplex was located rather near the atomizer surface. (It was actually the same as simplex A in Figure 23.) Because of the short half-life of atomic manganese in the sheath gas, its response surface probably has very similar slopes for horizontal and vertical displacements. For optimizations A and B, in Table 8, the simplex converged on an optimum a bit higher and farther in the negative horizontal direction than the cadmium optimizations, but convergence was achieved in a comparable number of

Table 8. Results of the Optimizations of Horizontal and Vertical Positions for Maximum Integrated Absorbance for Manganese.

Optimization	HPOS optimum (mm)	VPOS optimum (mm)	Iterations Required	Precision Requested (realized)		Iterations/ Response Evaluation
				HPOS (mm)	VPOS (mm)	
A	- .54	.61	15	.5 (.23)	.5 (.50)	1
B	-1.2	1.1	13	.5 (.40)	.5 (.33)	1
C	.27	.25	36	.5 (.40)	.5 (.09)	1
D	- .52	.42	15	.5 (.74)	.5 (.42)	3

iterations. Perusal of the progress of optimization C indicates that it contracted too much initially, and noise problems made it difficult for it to find the true optimum. In optimization D, three sample determinations were averaged for each response evaluation. This should decrease the susceptibility of the simplex algorithm to random noise. After 15 iterations the atomizer suffered catastrophic failure from aging, and this occurred before the true optimum was located. From the standpoint of achieving the most efficient optimization, increasing the number of runs per response evaluation does not help. The simplex algorithm is self-correcting for random errors if they are not so overwhelming that they cause premature contraction (49).

In the manganese optimizations, the algorithm converged on the vertical optimum more rapidly than it converged on the horizontal optimum in three of four optimizations. For cadmium, the convergence on the horizontal optimum was always rapid. This difference is probably due to the greater vertical slope in the manganese response surface than in the cadmium response surface. The overall performance of the simplex algorithm in finding an optimum close to the global optimum was poorer for manganese than for cadmium, which seems to indicate that there is more noise on the manganese response surface than on the cadmium response surface. Since the

atomization temperature for manganese is about 1000°C higher than the atomization temperature for cadmium, the disturbance of the sheath gas flow laminarity near the atomizer is greater for manganese, and undoubtedly this is at least part of the cause for the noisier response surface.

D. FOUR PARAMETER OPTIMIZATIONS

The labor-saving characteristics of the simplex algorithm as compared to grid-search optimizations become more significant as the dimensionality of the optimization is increased. To collect data to evaluate the entire response surface for a four factor optimization is too time consuming even for a totally automated system, such as the EAAA instrument. Since the operation of the algorithm in a four parameter optimization was untested, it was necessary to test the technique on four parameters whose effects on the selected response were either known or could be easily deduced.

1. Four Parameter Optimization of the Integrated Absorbance of Cadmium

The four factors chosen were the horizontal position, the vertical position, the amount of sample, and the length of the time delay between sample deliveries. The

chosen response was again the cadmium integrated absorbance. There is not expected to be any interaction between the horizontal and vertical position variables and the two new factors, so the horizontal and vertical optima should be the same as in the two parameter optimization. Obviously, the optimum value for the amount of sample solution delivered should be the maximum amount allowed by the upper boundary, and the length of the time delay between sample determinations should have no effect on the response.

In the five test cases shown in Table 9, the starting simplexes were located randomly in factor space according to the guidelines established in the previous section. Note that the number of iterations required for convergence was significantly greater than the number required in a two parameter optimization. Even though the delay time has no effect on the response chosen (integrated absorbance), in each case the simplex algorithm converges on an "optimum" value for this factor. The inconsequential nature of this factor is signalled by the totally random values on which the simplex converges for this parameter. This does demonstrate that the technique can find a true optimum even if one of the chosen factors has no effect on the response. The horizontal positions designated at the optimum are not unlike those found in the two parameter optimization, except that optimizations B and C yielded

Table 9. Results of Optimizations of Delay Time, Horizontal Position, Vertical Position, and Amount of Sample for Maximum Integrated Absorbance for Cadmium.

Optimization	DELT (sec)	HPOS (mm)	VPOS (mm)	AMNT (µℓ)	Iterations Required
A	9.3	0.0	.64	3.93	44
B	18.0	-.63	3.6	3.97	31
C	5.4	-.51	.87	3.94	39
D	15.3	.28	.38	3.89	28
E	9.0	.11	1.1	3.92	30

a result a little farther from the true global optimum than any of the optima found in the two parameter optimizations. This effect is even more pronounced in the optimization of the vertical parameter where one of the "optimum values" found was over 3 mm from the true optimum value. The approach of the sample amount to the upper boundary level is consistent throughout the test as was initially expected. Since this parameter has a very strong effect on the response, the convergence precision for this parameter is very good.

For optimization E, the Figures 27, 28A, 28B, 29A and 29B are progress plots of the response, vertical position, horizontal position, sample amount, and delay time, respectively. Comparison with parameter progress plots for the two dimensional optimization show that the improvement in response is more gradual, and the movement toward the horizontal and vertical optima is also more gradual. Also, the approach to the vertical and sample amount optima, which are located at the parameter boundaries, is rather slow and tentative. The difficulty that this simplex algorithm has in locating an optimum located at a boundary seems to be characteristic of this algorithm. The super modified simplex is thought to approach optimum located at a boundary value much more readily (37).

Unfortunately the progress plot of the delay time parameter contains no distinctive characteristics which

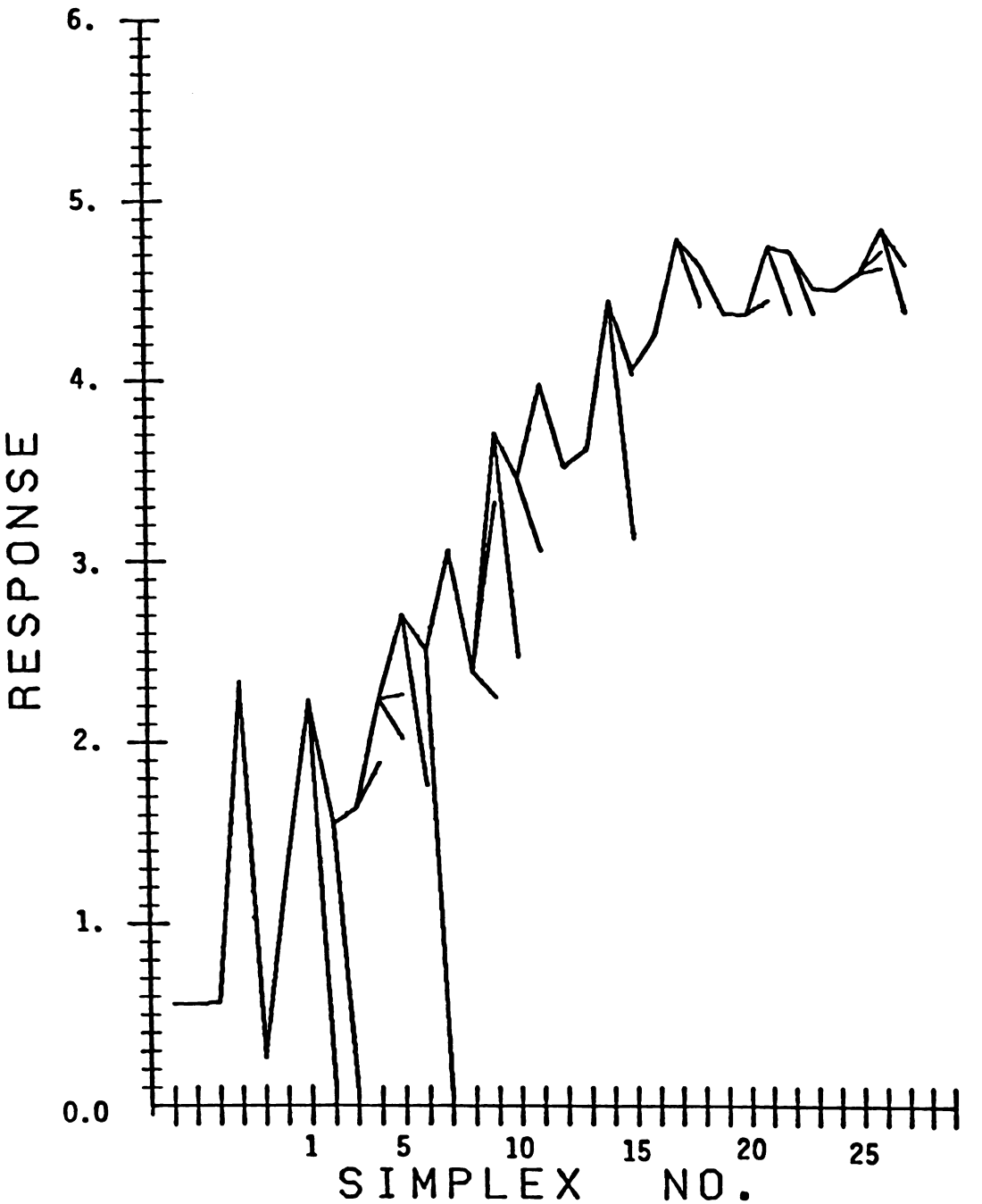


Figure 27. Progress Plot of the Cadmium Integrated Absorbance Response in a Four Parameter Optimization.

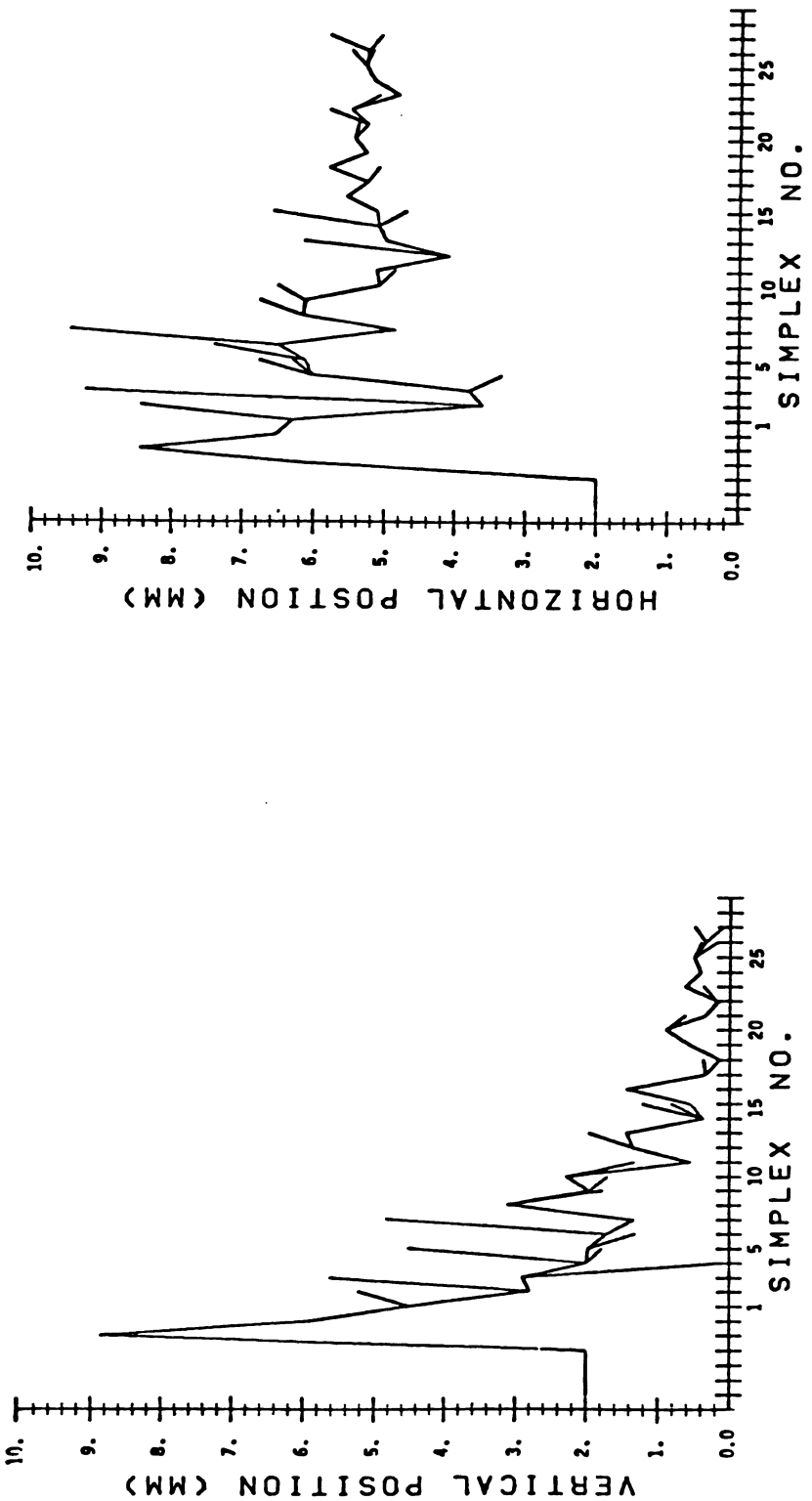


Figure 28. Progress Plots of the Vertical Position and the Horizontal Position in a Four Parameter Optimization of the Cadmium Integrated Absorbance.

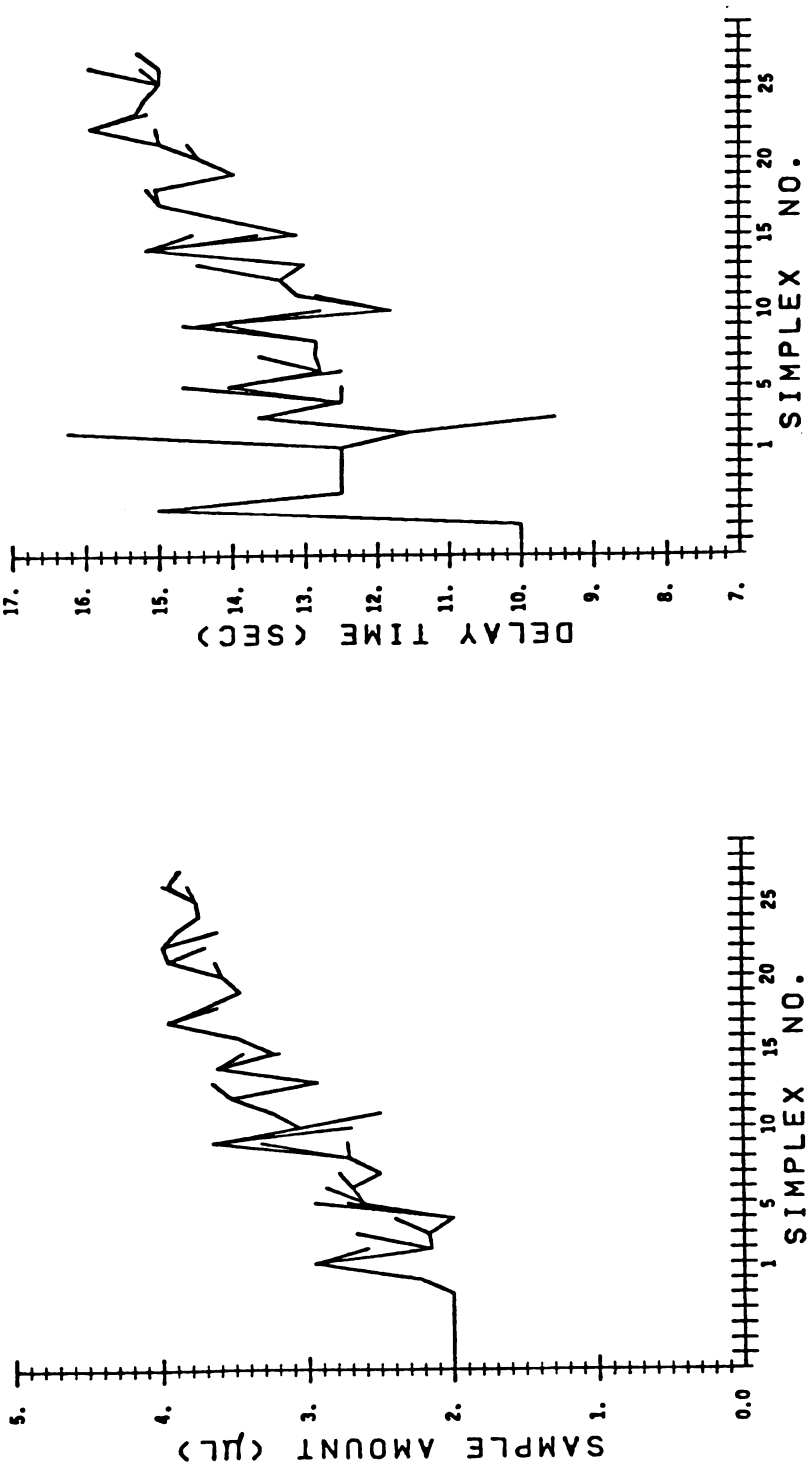


Figure 29. Progress Plot of the Sample Amount and the Delay Time in a Four Parameter Optimization of the Cadmium Integrated Absorbance.

would point out that this parameter has no effect on the response chosen. It appears that the only way to show definitively the insignificance of this factor is to compare the optimum values for this parameter in several optimizations.

2. Simplex Optimization with a Complex Response Function

The second four parameter optimization attempted was of a more practical nature. It is generally desirable in EAAA determinations, as with many other analytical determinations, to maximize the accuracy and precision of the measurement process and to minimize the analysis time. This means that the value used as feedback to the simplex algorithm must be some type of weighted sum of individual responses, which is representative of the analysis time and the precision and accuracy of the measurement. Within this context, the parameters chosen as factors for the four parameter optimization were horizontal position, vertical position, desolvation time, and desolvation power. Since a major portion of the analysis time under normal operating conditions is spent in desolvating the sample, minimization of this variable could represent a significant increase in instrument throughput. The desolvation power was chosen as a factor because it interacts with the desolvation time. That is, the time necessary to

desolvate the sample is not independent of the desolvation heating temperature.

The response chosen to measure the analysis time is straightforward. However, an experimentally practical measure of the measurement precision and accuracy is more difficult. In normal practice the standard deviation is used as a measure of the precision and an indication of the accuracy. The number of experiments necessary to give a reasonably reliable estimate of the standard deviation would cause the optimization to take a prohibitively long time. If one assumes that the noise in the region of the global optimum is homoscedastic and that the EAAA measurement is well-behaved with respect to the relationship of the integrated absorbance to the cadmium concentration, then the assumption that the magnitude of the integrated absorbance is representative of the precision and accuracy of the measurement is justified.

Rules for designing the function which calculates the combinational response from two individual responses have been established (34,47). However, each particular system requires some prior knowledge and careful consideration before a satisfactory response function can be established. First, it was necessary to write an equation for each individual component of the response function that would normalize each individual response to be near unity at its optimum value. For the signal response contribution,

Ra, the integrated absorbance was divided by the estimated maximum value of the integrated absorbance. Under these conditions,

$$Ra = \frac{ABSINT}{3.} \qquad (1)$$

The time response contribution, Rt, was normalized with respect to the upper and lower boundary constraints as follows.

$$Rt = \frac{1. - (DEST - 5.)}{40. - 5.} \qquad (2)$$

The combinational response, Rc, was then calculated as a weighted sum of these normalized responses by Equation 3,

$$Rc = .75Ra + .25Rt \quad . \qquad (3)$$

A subroutine was written to execute the calculation of this response function. The routine was assimulated into the load module for GBMDS2 program, and activated in the simplex mode of operation by setting the variable which indicates an alternate response evaluation will be required. After 40 iterations, the simplex algorithm had converged to the expected precision for the horizontal position (-0.45 mm), the vertical position (0.46 mm), and

the desolvation power (.46 volts at the DAC which controls the braid temperature). At this point the average desolvation time for the simplex was 16.4 s. But, the range of values for the vertices in the final simplex was 13.9 to 20.3 s.

The optimization was not totally successful. The shape of the response surface under this weighting scheme allowed a proper convergence for the horizontal and vertical parameters, but not for the other two factors. When a combinational response is used, the response surface may be thought of as the convolution of the individual response surfaces. This process will make it likely that the combinational response surface will be less regular than most individual response surfaces. Thus, the probability of a successful optimization is diminished. In this particular case, it appears that the weighting of the response toward the signal magnitude initially favored finding the location of the horizontal and vertical optima. Recall that it has already been pointed out that the simplex will converge on an "optimum" for a factor even if this factor is insignificant. In this weighting scheme, the effect of changes in the delay time were effectively screened by the large effect the position variables had on the calculated response. When the position variables had converged, the size of the simplex had diminished so much that it was entirely located on a plateau in the

response surface. This caused experimental noise to mask the impact that changes in the desolvation parameters had on the response. Changing the weighting factors to favor the desolvation parameters over the position parameters might allow the algorithm to find the true optimum. From this study, it appears that the parameters that are least significant and cause plateau regions in the response surface should be weighted most heavily so that they control the initial movements of the simplex. Thus their optima will be located before the simplex contracts significantly. This is not a panacea, because the portion of the response surface traversed by the simplex is dependent on both the starting simplex and the random effects of instrumental noise. For this reason, weighting schemes that might work well in one trial may fail miserably if the optimization is restarted with a different or even the same initial simplex. The super modified simplex algorithm might achieve more consistent results in this type of optimization since it has been suggested that it is immune to noise (37).

It has been shown in this section that four parameter optimizations are possible on the EAAA instrument if the response surface is well-behaved. It was also shown that combination response functions add a degree of complexity that increases the probability of failure of the simplex algorithm.

E. OPTIMIZATIONS OF THE MEASUREMENT PRECISION

1. Direct Optimization of the Measurement Precision

As stated in the previous section it would be desirable to optimize the precision of the EAAA measurement directly. However, this would require a large number of experimental observations. For this reason, only a three dimensional optimization was attempted. Three parameters were chosen which were thought to have a significant effect on the measurement precision: the vertical height, the sheath gas flow rate, and the atomization power. The vertical position should affect the precision by several mechanisms. The higher the observation window, the smaller the expected signal and the less laminar the sheath gas flow. Both contribute adversely to the precision. On the other hand, low observation heights increase the noise by increasing the amount of shot noise from the observation of background radiation from the atomizer. At very low positions, the atomizer may occlude some of the source radiation and increase the detrimental effect of some noise sources that are independent of the level of source radiation. High atomization temperatures can also contribute to the amount of continuum radiation observed by the system, and more tepid temperatures can cause incomplete atomization of the analyte. The sheath gas flow rate should influence the measurement noise at

both low and high flow rates, since the flow laminarity is destroyed at either extreme.

This optimization was actually carried out with a multi-minicomputer system. One minicomputer controlled the EAAA instrument and another executed the simplex routine. Response and test conditions were transferred between computers by the Pals asychronous link (i.e., the operator typed the response or test condition produced by one computer into the other computer's terminal). The operator was also required to adjust manually the gas flow rate. The starting simplex was generated by the simplex program. Four sample runs were taken for each determination of the measurement precision, and after 14 iterations, which required 120 sample runs, the simplex was halted. The range of values for the starting simplex and final simplex vertices are shown in Table 10. The flow rate increased, the height of the vertical observation window increased, but the atomization temperature remained essentially unchanged. The improvement in overall measurement precision was not very significant.

The failure of this optimization to converge is likely due primarily to the statistical noise in the response measurement. That is, the variance in the variance predicted for a set of only four determinations. The response is not only quite noisy, it also contains a plateau in the region of the optimum for at least the sheath gas

Table 10. Results of Optimization of the Gas Flow Rate, Vertical Position, and Atomization Power for Maximum Signal to Noise Ratio.

	Gas Flow Rate		Vertical Position		Atomization Power		S/N	
Simplex	Average	Range	Average	Range	Average	Range	Average	Range
Initial	2.6	2.2-3.0	7.8	7.0-9.4	5.1	4.7-6.1	3.4	1.6-6.
Final	3.3	3.1-3.5	10.6	9.2-11.6	5.1	4.6-6.0	4.5	3.8-5.1

flow parameter. Later studies showed that over the range of about 2.5 to 3.5 l/min, the flow rate has little effect on the measurement precision. It would also have been better to use some intuition in the decision about the size and location of the initial simplex rather than accepting the simplex suggested by the computer routine. Also of some concern, but probably not important here, is the fact that some noise is resultant from sources not under rigid control. As noted in Chapter 4 this noise is in the range of 1 to 3% with some long term drift. This noise level should only limit the precision of the final determination of the optimum, and should not be serious enough to invalidate totally the optimization process.

2. The Determination of the Optimum Integration Time Window

The attempt to optimize measurement precision by varying the limits on the time window for the evaluation of integrated absorbance was a good deal less ambitious than the optimization just described and met with a great deal more success. In this optimization, the integration start time and stop time were varied under the constraint that the stop time always had to be later than the start time. Several data files were recorded that contained a chronological series of 200 absorbance data points for cadmium collected over a 1.0 s period. Each of these

data files contained 10 replicate sample runs. The precision was evaluated within the limits of each new integration time window suggested by the simplex algorithm on this data set so that no additional on-line data acquisition was necessary. The data in Table 11 clearly show that optimizing the integration limits results in a significant increase in measurement precision only when the amount of analyte is near or below the detection limit found when absorbance points during the entire atomization step are integrated. Previous experiments in this genre, which had a peak detection algorithm, had not shown any statistically significant increase in the measurement precision (28). The problem with such algorithms is that it is very difficult to write one that will perform well for high and low sample concentrations, and that will perform with any consistency at or near the detection limit where the noise in the signal is quite significant. Furthermore, these algorithms can be expected to be advantagous only at low concentrations, where the noise sources that are independent of the photocurrent level become important.

Figure 30 is a typical absorbance peak shape for a 2 μl sample of 10 ppb cadmium, and is useful in the interpretation of the start and stop times of the signal integration listed in Table 11. To test the validity of the values found in these optimizations a scheme was devised in which the optimum time values found by the simplex

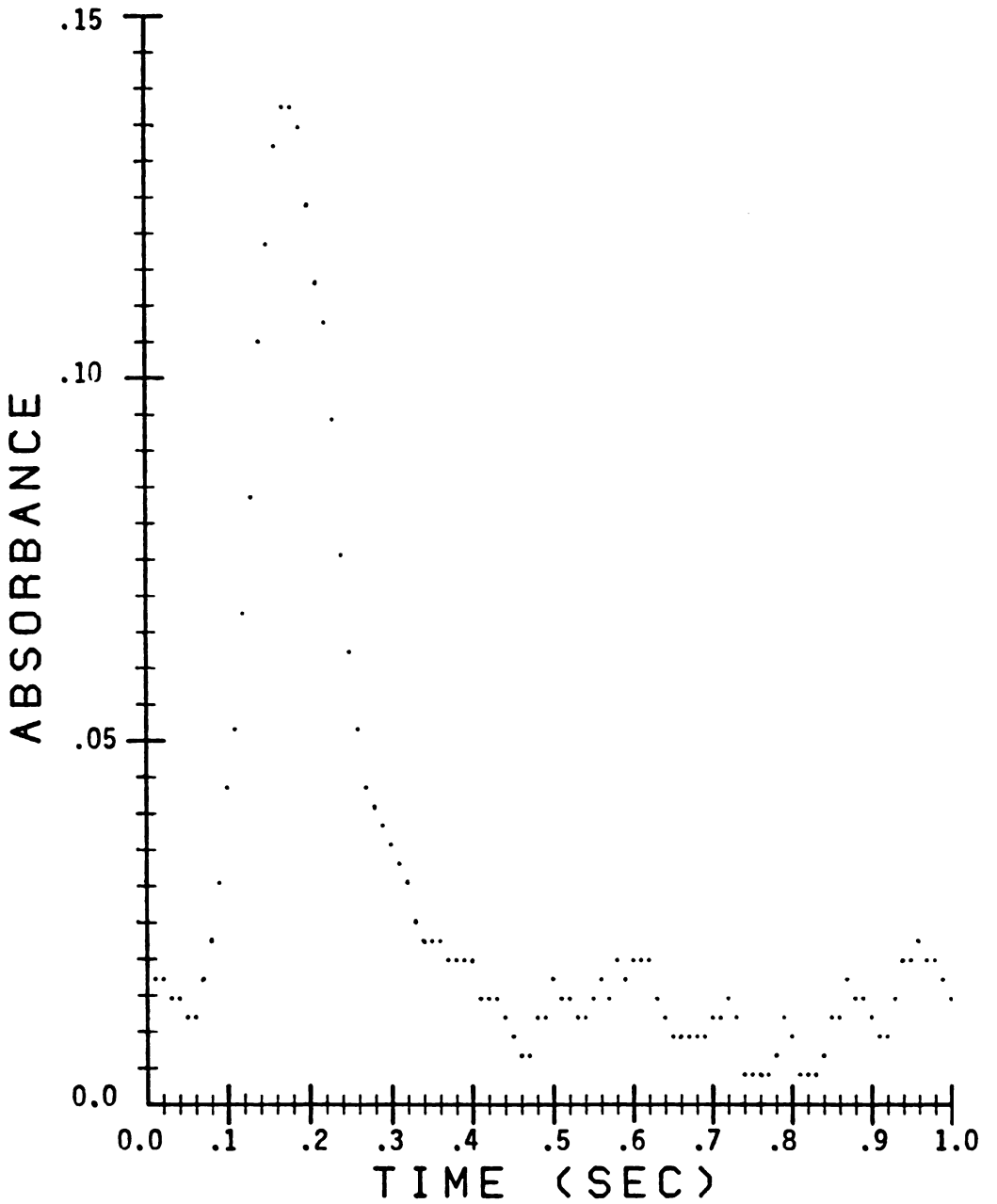


Figure 30. A Typical Absorbance Peak for 2 µl of 1 ppm Cadmium Solution

Table 11. Results of Optimizations of Integration Start and Stop Times.

Learning Set	Cd Conc.	Integration Times (s)		Precision (%RSD)	
	(ppb)	Start	Stop	At Optimum	Normal Integration
A	5000	.10	.43	2.0	2.7
B	100	.10	.31	2.2	2.4
C	10	.05	.19	10.5	27.
D	10	.14	.16	12.5	128.
E	10	.12	.18	12.5	118
F	10	.10	.18	19.6	77
G	10	.12	.19	19.2	63

* % RSD for the determination when the limits of integration are the beginning and th the end of the atomization heating period.

optimization were applied to several test sets. Table 12 contains the results of these tests. In Table 12, the learning sets were data sets on which the simplex optimization was performed, and the test sets were data sets which were collected under identical experimental conditions. The optimum time window as determined by simplex optimization of each of the learning data sets was then used to evaluate the data in the test sets. Also shown in Table 12 is the precision for each of the test data sets when the absorbance integration includes the entire atomization period. A 5 to 10 times improvement in the measurement precision resulted when the test data were evaluated between the simplex suggested limits as compared to the integration over whole atomization period. Also, the improvement in precision does not seem to vary significantly for a given test set when the integrated absorbance is evaluated in the different time windows found by the simplex optimization of each of the learning data sets.

The results of these optimizations can only be applied to a limited extent to actual determinations for several reasons. The width of the absorbance peak is related to the amount of the analyte in the sample so that a constant time window will observe a smaller portion of large peaks than of small peaks. This will cause curvature in calibration plots, but this is not an unreasonable price to pay for the observed increase in measurement precision

Table 12. Evaluation of Applicability of Simplex Determined Integration Time Window to Other Data Sets.

Learning Set	Test Set (%RSD) F	G	H	I	Normal Integration*
A	13	15	19	17	130
B	16	13	15	14	120
C	20	20	20	22	120
D	17	20	21	20	80
E	13	13	12	13	60

* % RSD for determination when the limits of integration are the beginning and end of the atomization heating period.

and sensitivity. The width and time location of the absorbance peak is also a function of the ashing and atomization temperatures. The atomizer will reach the final atomization temperature earlier if the ash temperature is high, so that the absorbance peak will occur earlier in time. Also the rate at which analyte atoms leave the atomizer is temperature dependent so that the width of the peak depends on the final atomization temperature.

These results point out the value of limited integration of the absorbance peak. Because of the interaction of the integration start and stop times with the ash temperature, atomization temperature, and sample amount, it is highly desirable to have these parameters available for optimization. These results lead to the conclusion that an on-line optimization of the measurement precision would have a very good chance of success. However, the number of experimental data points needed would be large and the acquisition of this data would be prohibitively time consuming in anything but a totally automated environment.

F. CONCLUSIONS AND PERSPECTIVES

The software required to implement the simplex technique on the EAAA instrument was neither complex nor lengthy, and the wetware (operator knowledge) required

was significant, but not unreasonable. The unique features of this work include: 1) the simplex optimization technique has never been applied to EAAA spectroscopy, 2) this is the first on-line application of the simplex that allowed the user any flexibility in the choice of parameters to be varied in the optimization. The on-line optimization of the integrated absorbance signals of cadmium and manganese proved that the technique was an efficient method of finding the global optimum if and only if a judicial choice of the initial simplex was made. The optimization of the integrated absorbance signal by varying the delay time, sample amount, horizontal position, and vertical position demonstrated the tendency of the simplex to converge on an "optimum" value for a factor even if it is insignificant. Restarting the simplex in another region of the response surface not only helps to decide if the optimum located was a local optimum or the global optimum, but will also point out the insignificance of a factor by converging on totally random optima for that parameter. The attempted optimization of a combinatorial response function demonstrated that as the response function becomes more complex, the response surface is likely to take a shape less favorable for the operation of the simplex algorithm. Optimization of the measurement precision was shown to be an experimentally difficult task because a good estimate of the variance requires a large

population of individual measurements. It would not be difficult to set up the optimization of the signal-to-background ratio, since the background value could be easily evaluated from the array which contains the data for the correction of atomizer continuum background radiation. The problem with this type of response function evaluation is that the HCDT source drift causes an imprecise determination of the background level. It would be more desirable to evaluate the standard deviation of this set of noise measurements and assume that this is a good approximation of the noise in the signal measurement. This procedure is still somewhat susceptible to HCDT drift, but software could be generated to minimize the effect of this drift. A direct optimization of the precision of the measurement would be very valuable and could be applied to the determination of practical samples such as the determination of manganese in chlorophyll (86).

There have been at least as many suggested improvements to the simplex optimization technique as there have been researchers who have used the technique. Most suggestions have been directed toward the desensitization of the algorithm to the effects of noise on the response surface (37,58,63). This work has pointed out the critical nature of the size, location, and orientation of the starting simplex on the success of the optimization. I propose

that the starting simplex be generated by selection of the sum of the vertices of a regular polyhedron of n+m dimensions, where n is the number of factors to be optimized and m is an arbitrary integer greater than 1. The response at each vertex of this figure is then evaluated, and the n best vertices and the worst vertex retained. These are then used to form the actual starting simplex for the simplex algorithm. In two dimensions this preliminary figure might be an equilateral pentagon. Normally the two best vertices will be adjacent vertices and the worst vertex will be opposite the face formed by the two best vertices. When the two next worse vertices are discarded, the remaining figure is a triangle elongated in the direction of the optimum. Thus the simplex can be moved rapidly toward the optimum on the first move. Often the region in which the simplex starts is the noisiest section of the response surface that it will have to traverse. Thus, it is critical to move it rapidly in the correct direction, so that the simplex does not fall prey to its natural statistical tendency to contract prematurely in the presence of noise in the response. The temptation to retain more than n+1 vertices in the working simplex should be restrained, since increasing the number of vertices will make the simplex move more sluggishly over the response surface.

REFERENCES

References

1. A. Walsh, Australian Patent No. 23041 (1953).

2. A. Walsh, Spectrochem. Acta, 7, 108 (1955).

3. C. T. J. Alkemade and J. M. W. Milatz, Appl. Sci. Res. 4B, 289 (1955).

4. C. T. J. Alkemade and J. M. W. Milatz, J. Opt. Soc. Am., 45, 583 (1955).

5. W. H. Wollaston, Phil. Trans. Roy. Soc. London, Ser A, 92, 365 (1802).

6. T. T. Woodson, Rev. Sci. Instr., 10, 308 (1939).

7. B. V. Lvov, Spectrochim. Acta., 17, 761 (1961).

8. R. Woodriff and G. Ramelow, Spectrochim. Acta., 23B, 665 (1968).

9. R. Woodriff and R. Stone, Appl. Optics, 7, 1337 (1968).

10. R. Woodriff, R. W. Stone, and A. M. Held, Appl. Spectrosc., 22, 408 (1968).

11. R. Woodriff, B. R. Culver, and K. W. Olsen, Appl. Spectrosc., 24, 530 (1970).

12. R. Woodriff and D. Shrader, Anal. Chem., 43, 1918 (1971).

13. G. K. Pagenkopf, D. R. Neuman, and R. Woodriff, Anal. Chem., 44, 2248 (1972).

14. R. Woodriff, B. R. Culver, D. Shrader, and A. B. Super, Anal. Chem., 45, 230 (1953).

15. H. Massmann, Spectrochim. Acta., 23B, 215 (1968).

16. H. Massmann, in Flame Emission and Atomic Absorption Spectrometry, Vol. 2, J. A. Dean and T. C. Rains, Eds., Marcel Dekker, New York, 1971.

17. Perkin-Elmer Corporation, Instrument News, 21, 4 (1970).

18. H. L. Kahn and S. Slavin, A. A. Newsletter, 10, 125 (1971).

19. Perkin-Elmer Corporation, Analytical Methods for Atomic Absorption Spectroscopy Using the HGA Graphite Furnace, Norwalk, Conn., March 1973.

20. T. S. West, and X. K. Williams, Anal. Chim. Acta., 45, 27 (1969).

21. R. G. Anderson, I. S. Maines, and T. S. West, Anal. Chim. Acta, 51, 355 (1970).

22. J. F. Alder and T. S. West, Anal. Chim. Acta, 51, 365 (1970).

23. K. W. Jackson, T. S. West, and L. Balchin, Anal. Chem., 45, 249 (1973).

24. A. Montaser and S. R. Crouch, 25th Pittsburgh Conf. Anal. Chem. Appl. Spectrosc., Papers 199 and 116 (1974).

25. A. Montaser, S. R. Goode, and S. R. Crouch, Anal. Chem., 46, 599 (1974).

26. A. Montaser and S. R. Crouch, Anal. Chem., 46, 1817 (1974).

27. A. Montaser and S. R. Crouch, Anal. Chem., 47, 38 (1975).

28. D. N. Baxter, Ph.D. Thesis, Michigan State University, East Lansing, MI, 1977.

29. A. Montaser, Ph.D. Thesis, Michigan State University, East Lansing, MI, 1974.

30. S. Augusta, CRC Crit. Rev. Anal. Chem., 4, 155 (1974).

31. G. E. P. Box, Biometrics, 10, 16 (1954).

32. G. E. P. Box and K. B. Wilson, J. Roy. Statist. Soc. B., 13, 1 (1951).

33. G. E. P. Box, Appl. Statisc., 6, 81 (1957).

34. C. W. Lowe, Trans. Instr. Chem. Engrs., 42, T334 (1964).

35. W. Spendley, G. R. Hext, and F. R. Hinsworth, Technometrics, 4, 441 (1962).

36. J. A. Nelder and R. Mead, Computer J., 7, 308 (1965).

37. M. W. Routh, P. A. Schwartz, and M. B. Denton, Anal. Chem., 49, 1422 (1977).

38. K. W. Lam, Clin. Chem., 23, 89 (1977).

39. D. J. Legget, Anal. Chem., 49, 276 (1977).

40. J. K. Haken, M. S. Wainwright, and R. J. Smith, J. Chromat., 133, 1 (1977).

41. E. Baumgarten, F. Weinstrauch and H. Hoffkes, J. Chromat., 138, 347 (1977).

42. G. L. Ritter, S. R. Lowry, C. L. Wilkins, and T. L. Isenhour, Anal. Chem., 47, 1951 (1975).

43. T. R. Brunner, C. L. Wilkins, T. F. Lam, L. J. Soltzberg, and S. L. Kaberline, Anal. Chem., 48, 1146 (1976).

44. D. E. Long, Anal. Chim. Acta, 46, 193 (1969).

45. S. N. Deming and S. L. Morgan, Anal. Chem., 45, 278A (1973).

46. S. N. Deming and P. G. King, Research/Development, 25, 22 (1974).

47. S. L. Morgan and S. N. Deming, Anal. Chem., 46, 1170 (1974).

48. P. G. King, S. N. Deming, and S. L. Morgan, Anal. Lett., 8, 369 (1975).

49. M. J. Holue, D. E. Long, and D. Smette, Anal. Lett., 3, 401 (1970).

50. F. P. Czech, J. Ass. Offic. Anal. Chem., 56, 1489 (1973).

51. F. P. Czech, J. Ass. Offic. Anal. Chem., 56, 1496 (1973).

52. L. R. Parker, Jr., S. L. Morgan, and S. N. Deming, Appl. Spectrosc., 29, 429 (1975).

53. R. D. Krause and J. A. Lott, Clin. Chem., 20, 775 (1974).

54. W. K. Dean, K. J. Heald, and S. N. Deming, Science, 189, 805 (1975).

55. F. Darvas, J. Med. Chem., 17, 799 (1974).

56. D. E. Fonner, J. R. Buck, and G. S. Banker, J. Pharm. Sci., 59, 1587 (1970).

57. W. E. Rippetoe, E. R. Johnson, and T. J. Vickers, Anal. Chem., 47, 436 (1975).

58. R. R. Ernst, Rev. Sci. Instr., 39, 998 (1968).

59. E. R. Johnson, C. K. Mann, and T. J. Vickers, Appl. Spectrosc., 30, 415 (1976).

60. P. G. King, Ph.D. Thesis, Emory University, Atlanta (1974).

61. Q. V. Thomas, L. Kryger, and S. P. Perone, Anal. Chem., 48, 761 (1976).

62. M. W. Routh, P. A. Swartz, and M. B. Denton, Anal. Chem. 49, 1422 (1977).

63. E. R. Johnson, Ph.D. Thesis, Florida State University, Tallahassee, Florida (1975).

64. K. M. Cellier, and H. C. T. Stace, Appl. Spectrosc., 20, 26 (1966).

65. M. L. Parsons and J. D. Winfordner, Appl. Spectrosc., 21, 368 (1967).

66. N. V. Mossholder, V. A. Fassel, and R. N. Knisley, Anal. Chem., 45, 1614 (1973).

67. J. L. Malakoff, J. Ramirez-Munoz. and A. Scott, Anal. Chim. Acta, 42, 515 (1968).

68. S. R. Crouch, D. N. Baxter, E. H. Pals, and E. R. Johnson, Anal. Chem., 50, 291A (1978).

69. Digital Equipment Corp., OS/8 Handbook, Maynard, Mass. Digital Equipment Corp., 1974, Chapter 8.

70. S. P. Keller and J. F. Eagleston, J. Chem. Ed., 48, 317 (1971).

71. E. C. Toren, Jr., R. N. Carrey. A. E. Sherry, and J. E. Davis, Anal. Chem., 44, 339 (1972).

72. H. E. Keller, G. E. Courtois, and J. E. Keller, Chem. Instr., 4, 269 (1972).

73. G. S. Cembrowski, D. B. Cottrel, and E. C. Torren, Computers and Chemistry, 1, 45 (1975).

74. R. E. Dessy, Anal. Chem., 49, 1104A (1977).

75. L. A. Leventhal, Killobaud, 11, 24 (1977).

76. F. J. M. Massen, F. D. Posma, J. Balke, Anal. Chem., 46, 1445 (1974).

77. V. Sacchetti, G. Tessari, and G. Torsi, Anal. Chem., 48, 1175 (1976).

78. C. J. Molnar and J. D. Winfordner, Anal. Chem., 46, 1807 (1974).

79. S. R. Goode, A. Montaser, and S. R. Crouch, Appl. Spec., 27, 335 (1973).

80. S. R. Crouch, A. Montaser, and S. R. Goode, in "Information Chemistry: Assisted Chemical Research Design", (S. Fujiwara and H. B. Mark, eds.), University of Tokyo Press, Tokyo, 1975, pp. 107-124.

81. "Sigma Stepping Motor Handbook", (Sigma Instruments, eds.), Braintree, Mass., 1972.

82. R. G. Anderson, I. S. Maines, and T. S. West, Anal. Chim. Acta, 51, 355 (1970).

83. D. N. Baxter, E. H. Pals, E. R. Johnson, and S. R. Crouch, 27th Pittsburgh Conf. Anal. Chem. Appl. Spectrosc., Paper 169, Cleveland, OH, March 1976.

84. R. D. Reeves, B. M. Patel, C. J. Molnar, and J. D. Winefordner, Anal. Chem., 45, 2446 (1973).

85. M. D. Joseph and E. R. Johnson, unpublished program, 1977.

86. J. T. Gano, E. H. Pals, and S. R. Crouch, Current Research (1978).

APPENDIX A

Selected Program Listings

```
C
C
C
C       PROGRAM FUNCTION:  MAIN PROGRAM FOR CBA EXPERIMENTS
C                          MODIFIED TO ALLOW SIMPLEX OPTIMIZATIONS
C       PROGRAMMER:  E. H. PALS
C
C       VERSION: *1A-1          (2.03)          (3.01)
C       DATE: 10/10/76          (12/31/76)      (8/1/77)
C       FILENAME: EPMAIN.F4     (EPMNV2.F4)     (EPMNS2.F4)
C
C       LOADING INSTRUCTIONS:  MUST BE LOADED INTO OVERLAY ZERO.
C
C       NOTE:  FURTHER INFORMATION IS AVAILABLE IN
C       THE FORM OF A HELP FILE.
C
C       CHANGES VERSION 2.01:
C       1.  DATA AND CALC ARE COMBINED INTO ONE PROGRAM.
C       2.  COMMON VARIABLE DEFINITIONS ARE UPDATED.
C       3.  ADD OPTION OF USER SUBROUTINE THREE
C
C       CHANGES VERSION 2.02:
C       1.  SWITCH REGISTER OPTION FOR PLOTTING IS ADDED
C
C       CHANGES VERSION 2.03:
C       1.  CBA SWITCH REGISTER OPTION 4 IS USED TO FORCE CALL
C           TO PLOT.
C
C       CHANGES VERSION 3.01:
C       1.  MODIFIED TO ALLOW SIMPLEX OPTIMIZATIONS.
```

```
C
C
C
C       PROGRAM FUNCTION:  MAIN PROGRAM FOR GBA EXPERIMENTS
C                          MODIFIED TO ALLOW SIMPLEX OPTIMIZATIONS
C       PROGRAMMER:  E. H. PALS
C
C       VERSION: *1A-1          (2.03)          (3.01)
C       DATE: 10/10/76          (12/31/76)      (8/1/77)
C       FILENAME: EPMAIN.F4     (EPMNV2.F4)     (EPMNS2.F4)
C
C       LOADING INSTRUCTIONS:  MUST BE LOADED INTO OVERLAY ZERO.
C
C       NOTE:  FURTHER INFORMATION IS AVAILABLE IN
C       THE FORM OF A HELP FILE.
C
C       CHANGES VERSION 2.01:
C       1.  DATA AND CALC ARE COMBINED INTO ONE PROGRAM.
C       2.  COMMON VARIABLE DEFINITIONS ARE UPDATED.
C       3.  ADD OPTION OF USER SUBROUTINE THREE
C
C       CHANGES VERSION 2.02:
C       1.  SWITCH REGISTER OPTION FOR PLOTTING IS ADDED
C
C       CHANGES VERSION 2.03:
C       1.  GBA SWITCH REGISTER OPTION 4 IS USED TO FORCE CALL
C           TO PLOT.
C
C       CHANGES VERSION 3.01:
C       1.  MODIFIED TO ALLOW SIMPLEX OPTIMIZATIONS.
```

```
C
C
C
C       METHOD:
C       1.  CALL INPUT- THIS ROUTINE ALLOWS INITIALIZATION
C           OF INSTRUMENT PATAMETERS.  THE ROUTINE WILL NOT
C           ALLOW EXIT UNLESS: ALL PARAMETERS ATE WITHIN
C           LEGAL LIMITS, A DARK CURRENT VALUE HAS BEEN TAKEN
C           THE SAMPLER AND POSITIONER INITAILIZED. IT ALSO
C           STARTS TE REAL TIME CLOCK.
C       2.  CALL FLOW- THIS ROUTINE HANDLES CHANGING THE
C           POSITION AND SAMPLE SPECIFICATIONS.  WHEN AN
C           ASSIGNED TASK HAS BEEN COMPLETED THE OPERATOR
C           MAY ENTER COMMENTS AND CLOSE THE FILE.  EXIT
C           MAY BE BACK TO MAIN WITH INSTRUCTIONS
C           TO GO EITHER TO THE DATA TAKING ROUTINE OR TO
C           TO GBA MONITOR(SUBROUTINE INPUT).
C       3.  CALL DATA- THIS PROGRAM EXECUTES CALLS TO POS,SAMPD,DAC,
C           ADC, AND CLOCK ROUTINES TO ACQUIRE READINGS OF 100%T,
C           SIGNAL ATTENUATED BY SAMPLE, AND BACKGROUND EMMISION OF GBA.
C           THE ABSORBANCE RESULTS ATE THEN CALCULATED, THE DATA
C           MAY THEN BE PACKED TO PASS TO THE PLOT ROUTINE, OR WRITTEN
C           ON ONE OR BOTH OF TWO OUTPUT FILES, AND/OR
C           OUTPUT TO THE LINE PRINTER.  (COMBINES FUNCTIONS OF
C           VERSION 1 PROGRAMS DATA AND CALC)
C       4.  IF REQUESTED CALL SUBTHR-  THIS IS AN OPTIONAL
C           USER SUBROUTINE.  IT IS NOT NECESSARY FOR CORRECT PROGRAM
C           EXECUTION AND IF IT IS NOT REQUESTED IT NEED NOT BE
C           LOADED INTO THE LOAD MODULE.
C       5.  IF REQUESTED CALL PLOT AND PLOT THE RESULTS
C           ON THE ADDS AND/OR THE LINEPRINTER.
C       6.  RETURN TO STEP2 TO FIND OUT IF TASK IS FINISHED.
```

```
C
C       PROGRAM MAIN EXECUTABLE CODE FOLLOWS
C
        COMMON PLTT,LPT,RAD,HDF,WAIT,DEST,ASHT,ATMT,DELT,DESP
        COMMON ASHP,ATMP,SCAN,VER,EOR,VERI,EORI,VERL,EORL
        COMMON NMSCAN,NMRUN,SAMP,SAMPSZ
        COMMON FIRST1,ISAMC,NSCAN,HPOS,VPOS,IRUN,ISAMP
        COMMON AMNT,ISAMCA,TAG,XIOIDK,XIDK
        COMMON FREQ,INTST,INTSP
        COMMON /A/ISAMPL(40),NRUN,IGOTO
        COMMON /B/DATA1(100),DATA2(100),AMIN,AMAX,IDIR
        COMMON /CBLK/MODE1,SUB1,SUB2,FILE1,FILE2,DARKT,PINIT,SINIT,SUB3
        COMMON /IDEB/ITER,VINIT(5,4),PM,ISVAR(4),ITYPE,LSMP,KRUN
        LOGICAL PLTT,LPT,RAD,HDF,WAIT,SCAN,SAMP,FIRST1,SUB3,LSMP
C       BEGIN BY CALLING INPUT TO SET UP PARAMETERS
1       CALL INPUT
C       NOW CALL SIMPLEX PROGRAM IF OPTIMIZATION REQUESTED
        IF(.NOT.LSMP) GO TO 2
        CALL SMEXOPT
        GO TO 1
C       NOW CALL FLOW-- IS THE TASK FINISHED, WHAT IS TO BE DONE NEXT?
2       CALL FLOW
        GO TO(1,1,3),IGOTO
C       CALL DATA EXERCISE THE HARDWARE
C       ALSO, CALCULATE THE RESULTS
3       CALL DATA
C       IF REQUESTED GO TO USER PROGRAM IN LEVEL ONE
        IF(SUB3) CALL SUBTHR
C       IF REQUESTED PLOT THE RESULTS
        IF(PLTT.OR.(SWITCH(4.).EQ.1.)) CALL POT
        GO TO 2
        END
```

```
C
C
C-      BLOCK DATA SUBROUTINE TO INITIALIZE COMMON FOR GBA
C-      PROGRAMMER: E. H. PALS
C
C-      VER:  2.10
C
C-      COMMENTS: AS VARIABLES ARE ADDED TO COMMON THIS PROGRAM MUST
C-      BE UP DATED PER PAGE 8-119 OS/8 MANUAL.
C
C-      LOADING REQUIREMENTS: THIS SUBPROGRAM MUST BE LOADED
C-      IN LEVEL ZERO AND CANNOT BE MADE PART OF A LIBRARY.
C
C
C       VARIABLE DESCRIPTION:
C
C       LOGICAL VARIABLES-
C       PLTT-    PLOT RESULTS
C       LPT-     OUTPUT RESULTS TO LINE PRINTER
C       RAD-     USE RADIATION PROGRAMING (VS. POWER PROG.)
C                DURING ATOMIZATION STEP
C       HDF-     USE HYDROGEN DIFFUSION FLAME DURING ATOMIZATION
C       WAIT-    PAUSE AFTER PLOTTING TO  ALLOW DATA INSPECTION
C                OF MANUAL DELIVERY FO SAMPLE
C       FIRST1-  USED TO INDICATE TO FLOW WHETHER INITIALIZATION OF
C                VARIABLES IS NECESSARY
C       SCAN-    A ONE OR TWO DIMENSIONAL SCAN IS REQUESTED
C       SAMP-    USE OF THE AUTO SAMPLER FOR SAMPLE DELIVERY IS REQUESTED
C       SUB1-    CALL USER SUBROUTINE ONE FROM DATA BEFORE CALCULATING
C                RESULTS
C       SUB2-    CALL USER SUBROUTINE TWO FROM DATA AFTER CALCULATING
C                RESULTS
C       SUB3-    CALL USER SUBROUTINE SUBTHR FROM MAIN AFTER RETURNING
C                FROM DATA
C       FILE1-   WRITE INTEGRATED ABSORBANCE, ETC. TO FILE SPECIFIED
C                AS DEVICE 5 TO FRTS
C       FILE2-   WRITE BACKGROUND CORRECTED ABSORBANCE FOR EACH
C                DATA POINT TAKEN
C       DARKT-   DARK CURRENT VALUE HAS BEEN TAKEN
C       PINIT-   POSITIONER HAS BEEN INITIALIZED
C       SINIT-   SAMPLER HAS BEEN INITIALIZED
C
C
C
C
C       REAL VARIABLES-
C
C       DELT-    DELAY TIME BETWEEN SAMPLE RUNS (SECS.)
C       DEST-    DESOLVATION TIME (S)
C       DESP-    DESOLVATION POWER (VOLTS FROM DAC OUTPUT)
C       ASHT-    ASH TIME (S)
C       ASHP-    ASH POWER (V)
C       ATMT-    ATOMIZATION TIME(S)
C       ATMP-    ATOMIZATION POWER(V)
C       VER-     INITIAL VERTICAL POSITION (MM)
C       HOR-     INITIAL HORIZONTAL POSITION (MM)
C       VPOS-    LAST (PRESENT) HORIZONTAL POSITION (MM)
C       HPOS-    LAST (PRESENT) VERTICAL POSITION (MM)
C       VERI-    WHEN SCANNING DISTANCE THAT VERTICAL POSITION IS
C                INCREMENTED (MM)
C       HORI-    AS ABOVE FOR THE VERTICAL POSITION (MM)
C       VERL-    MOST EXTREME LEGAL VERTICAL POSITION IN A GIVEN SCAN (MM)
C       HORL-    MOST EXTREME LEGAL HORIZONTAL POSITION IN A GIVEN SCAN (MM)
C       SAMPSZ-  VARIABLE USED TO INDICATE SAMPLE SIZE WHEN USING MANUAL
C                SAMPLE DEPOSITION
```

```
C       AMNT-    AMOUNT PT SAMPLE DELIVERED BY AUTOSAMPLER (UL)
C       XIDK-    AVERAGE OF 100 DARK CURRENT READINGS (0-4095)
C       XI0IDK-  AVERAGE OF 100 READINGS OF 100%T READONGS
C       ABSINT-  INTEGRATED ABSORBANCE FOR A SAMPLE RUN
C       FREQ-    DATA ACQUISITION RATE (HZ)
C       AMIN-    MINIMUM ABSORBANCE READING IN A PARTICULAR SAMPLE RUN
C       AMAX-    MAXIMUM ABSORBANCE IN A PARTICULAR SAMPLE RUN
C       DATA1-   ARRAY CONTAINS 100 TIME REPRESENTATIVE VALUES
C                FOR THE SAMPLE ATTENUATED TRANSMITTANCES
C       DATA2-   ARRAY CONTAINS 100 BACKGROUND TRANSMITTANCES
C-      TAG1-    CONTAINS A2 FORMAT TAG FOR DATA OUTPUT STATMENTS
C
C
C
C       INTEGER VARIABLES-
C
C       NMSCAN-  NUMBER OF SCANS REQUESTED
C       NSCAN-   NUMBER OF SCANS COMPLETED
C       NMRUN-   NUMBER OF RUNS REQUESTED
C       NRUN-    NUMBER OF RUNS COMPLETED
C       ISAMC-   SAMPLE SPECIFICATION PRESENTLY BEING EXECUTED
C       IRUN-    NUMBER OF RUNS REQUESTED FOR A PARTICULAR SAMPLE
C                NUMBER AND AMOUNT
C       ISAMCA-  NUMBER OF SAMPLE DELIVERIES COMPLETED ON THE PRESENT
C                SAMPLE  SPECIFICATION
C       ISAMP-   SAMPLE CUP NUMBER FROM AUTOSAMPLER THAT WAS REQUESTED
C       IGOTO-   CONTROLS PROGRAM FLOW IN MAIN UPON RETURN FROM SUBROUTINE
C       IDIR-    GIVES THE DIRECTION (POSITIVE OR NEGATIVE) OF THE
C                LAST HORIZONTAL MOVE
C       ISAMPL-  ARRAY CONTAINS SAMPLER SPECIFICATION WORDS EACH OF
C                WHICH IS A CODED AND PACKED WORD WHICH INDICATES
C                THE SAMPLE NUMBER, AMOUNT, AND NUMBER OF DELIVERIES
C                REQUESTED
C       MODE-    INDICATES DEVICE FROM WHICH INSTRUCTIONS ARE TO BE READ
C                0= ADDS TERMINAL
C                4= DEVICE (FILE) SPECIFIED AS NUMBER 4 TO FRTS
C
C
C
C       SIMPLEX VARIABLES-
C
C       LSMP-    LOGICAL VARIABLE INDICATES SIMPLEX MODE OF OPERATION
C       ITYPE-   INDICATES MODE OF CALCULATING RESPONSE
C       ISXVAR-  INTEGER ARRAY WHICH CONTAINS THE NUMBERS WHICH RELATE TO
C                WHICH VARIABLES ARE TO BE OPTIMIZED.
C       IDEM-    DIMENSIONALITY OF THE SPACE (NO. OF FACTORS)
C       V(I,J)-  VALUE OF FACTOR J AT VERTEX I
C       ALIM(3,J)-       LOWER UPPER LIMITS, PRECISSION IN EACH FACTOR J
C       IRET(I)-         NO. OF MOVES VERTEX I HAS BEEN RETAINED
C       IMV-     TYPE OF MOVE
C       IWRST-   INDEX OF NEW WORST POINT
C       INTST-   NUMBER OF POINT WHERE INTEGRATION STARTS
C       INTSP-   NUMBER OF THE POINT WHERE INTEGRATION STOPS
```

```
      BLOCK DATA
C-    VARIABLES USED BY MANY OF THE ROUTINES ARE IN BLANK COMMON
      COMMON PLTT,LPT,RAD,HDF,WAIT,DEST,ASHT,ATMT,DELT,DESP
      COMMON ASHP,ATMP,SCAN,VER,HOR,VERI,HORI,VERL,HORL
      COMMON NMSCAN,NMRUN,SAMP,TAG1
      COMMON FIRST1,ISAMC,NSCAN,HPOS,VPOS,IRUN,ISAMP
      COMMON AMNT,ISAMCA,ABSINT,XI0IDK,XIDK
      COMMON FREQ,INTST,INTSP
      COMMON /A/ISAMPL(40),NRUN,IGOTO
      COMMON /B/DATA1(100),DATA2(100),AMIN,AMAX,IDIR
      COMMON /CBLK/MODE,SUB1,SUB2,FILE1,FILE2,DARKT,PINIT,SINIT,SUB3
      COMMON /SPBLK/SP1,SP2,SP3,SP4,SP5,SP6
      COMMON /ISIM/ IDIM,V(5,4),R(5),ALIM(3,4),IRET(5),IMV,IWRST
      COMMON /IDEB/ITER,VINIT(5,4),PM,ISXVAR(4),ITYPE,LSMP,KRUN,IL
      DATA AMNT/1.0/
      DATA MODE/0/
C-    DECLARE LOGICAL VARIABLES AND INITIALIZE OPTIONS
      LOGICAL PLTT,LPT,RAD,HDF,WAIT,SCAN,SAMP,FIRST1,FILE
      LOGICAL SUB1,SUB2,FILE1,FILE2,DARKT,PINIT,SINIT,SUB3,LSMP
      DATA PLTT,LPT,RAD,HDF,WAIT,SCAN,SAMP/1,1,1,0,0,0,0/
      DATA FIRST1,SINIT,SUB3/1,0,0/
      DATA SUB1,SUB2,FILE1,FILE2,DARKT,PINIT,LSMP/0,0,0,0,0,0,0/
C-    INITIALIZE HEATING TIME PERIODS
      DATA DEST,ASHT,ATMT,DELT/10.,10.,.5,10./
C-    INTIALIZE POWERS
      DATA DESP,ASHP,ATMP/.25,.25,1.0/
C-    INITIALIZE SCAN PARAMETERS
      DATA VER,HOR,VERI,HORI,VERL,HORL/1.,0.,1.,1.,12.,4./
      DATA NMRUN,NMSCAN/1,1/
C-    INITAILIZE DATA TAKING FREQUENCY TO 200 HZ
      DATA FREQ,ABSINT/200.,0./
C     INITIALIZE OTHER VARIABLES
      DATA VPOS,HPOS,NRUN,ISAMP,AMNT/0.,0.,1,1,2./
      DATA INTST,INTSP/1,50/
      DATA IGOTO/3/
      DATA SP1,SP2,SP3,SP4,SP5,SP6/6*0./
      DATA TAG1/'DA'/
      END
```

```
C
C
C
C       PROGRAM NAME: SUBROUTINE INPUT
C       VER:  3.01
C
C       PROGRAMMER: E. H. PALS
C                CHEMISTRY DEPT., MSU
C                E. LANSING, MICH 48824
C
C       CALLING FORM: CALL INPUT
C
C       CALLING REQUIREMENTS:  ALL GBA VARIABLES MUST BE STORED
C                IN COMMON AND THE INITIAL VALUES DEFINED IN
C                THE BLOCK DATA SUBROUTINE.
C
C       LOADING REQUIREMENTS: MUST BE LOADED IN LEVEL 1.
C
C       EXTERNAL ROUTINES: EPLIMT,ANYCHR,EPBKV2,EPSAP,EPSAMP
C                EPPOS,CLOCK,EPDAC,EPSPV2,EPDARK,EPADC
C                EPAD8M,EPPOSD,EPSAPD,PARINS
C
C       VARIABLE DESCRIPTION: SEE PROGRAM MAIN AND BLOCK DATA SUBROUTINE
C
C       PURPOSE:  THIS SUBROUTINE IS ENTERED WITH A DEFAULT SET
C       OF PARAMETERS WHICH MAY BE CHANGED AS DESIRED BY SETS OF
C       COMMANDS 2 TO 6 CHARACHTERS IN LENGTH, SOME OF WHICH
C       REQUIRE A NUMERICAL ARGUMENT.  UPON COMPLETION CONTROL
C       IS PASSED BACK TO THE MAIN PROGRAM.
C       A MORE COMPLETE DESCRITION OF THE COMMAND SET FOLLOWS.
C
C       CHANGES VERSION *1D:
C       1.  COMMON DEFINITIONS ARE CHANGED TO COINCIDE WITH
C           THE COMMON DEFINITIONS IN THE OTHER SUBROUTINES.
C       2.  COMMANDS ARE ADDED WHICH CAUSE THE SAMPLER TO
C           BE INITAILIZED.
C       3.  INPUT SPECIFICATIONS FOR SAMPLER DELIVERY ACTIONS
C           MAY NOW BE ENTERED.
C       4.  THE REAL TIME CLOCK IS STARTED IN THIS
C           ROUTINE BEFORE EXITING.
C       5.  ***NOTE*** NOT ALL THE COMMAND STRING
C           DOCUMENTAION AND HELP COMMAND LISTINGS HAVE BEEN
C           BROUGHT UP TO DATE (10/11/76).
C       CHANGES *1E:
C       1.  CHANGE ISAMPL TO SAPINT SUB NAME
C       2.  CORRECT TYPING ERRORS
C
C       CHANGES VERSION:  2.01
C       1.  COMMANDS ADDED TO TAKE THE PLACE OF FILE SETUP SUBROUTINE.
C       2.  ADDED COMMANDS TO ALLOW BATCH STREAMING OF INPUT
C           SPECIFICATIONS.
C       3.  ADDED COMMANDS TO ALLOW EXECUTION OF TWO USER SUBROUINES.
C       4.  ADDED COMMAND TO ALLOW OUTPUT OF COMMENT TO LINE PRINTER.
C       CHANGES VERSION 2.02:
C       1.  BATCH MODE COMMAND ERROR CAUSES MODE TO GO TO LOCAL
C       2.  DATE AND PROGRAM VERSION NUMBER ADDED TO OUTPUT FILES
C       CHANGES VERSIONS 2.03 AND 2.04:
C       1.  FILE OUTPUT FORMS ARE MODIFIED
C       CHANGES VERSION 2.06:
C       1.  MODIFIED TO READ GBA SWITCH REGISTER NOT COMPUTER SW REG.
C       CHANGES VERSION 2.07:
C       1.  FILE OUTPUT STRUCTURE CHANGED TO SHAPE =10G13.6
C       2.  BACKGROUND RUN DESOLVATE,DELAY STEPS MODIFIED
C       CHANGES VERSION 2.09:
C       1.  LIMIT CHECKS MOVED TO FLOW ROUTINE FROM INPUT.
```

```
C       CHANGES VERSION 2.10:
C       1.  COMMAND MSC AND / ADDED TO FACILITATE BATCH STREAM OPERATION
C       2.  COMMAND TO CHANGE FILE DATA TAG ADDED
C       CHANGES VERSION 3.01:
C       1.  EXPANDED TO INCLUDE SIMPLEX ROUTINES.
C
C
C
C       METHOD:
C
C       1.  PICK UP THE CURRENT DATE FROM THE SYSTEM
C       2.  ISSUE A PROMPT AND RING TERMINAL BELL
C       3.  READ A COMMAND STRING IN A6 FORMAT AND STORE AS VARIABLE "ARG"
C       4.  IF IN BATCH MODE ECHO COMMAND TO ADDS TERMINAL
C       5.  CHECK ARG VERSUS EACH INDIVIDUAL COMMAND POSSIBLE.
C           ARG EQUALS A LEGAL COMMAND?
C               YES-  EXECUTE THE REQUESTED TASK OR TASKS, GO TO 6
C               NO-  ECHO A QUESTION MARK AND THE ARG, GO TO 2
C       6.  WAS ARG THE GO COMMAND?
C               NO-   GO TO 2
C               YES-   A.  IS THERE A PARAMETER LIMIT VIOLATION?
C                              NO-  GO TO B
C                              YES- PRINT ERROR MESSAGE, GO TO 2
C                          B.  HAVE POSITONER, DARK CURRENT, AND
C                              AUTOSAMPLER BEEN INITIALIZED?
C                              NO-  PRINT ERROR MESSAGE GO TO 2
C                              YES-  GO TO C
C                          C.  SCANNING?
C                              NO-  SET VARIABLES SO THAT FLOW RECOGNIZES
C                                   NOT SCANNING
C                              YES- GO TO D
C                          D.  PRINT LINE PRINTER HEADING IF REQUESTED (LPT)
C                          E.  SET CLOCK RATE AND CLEAR TIME TO 0 SEC
C                          F.  WAIT FOR POSITIONER TO REACH DESTINATION
C                              THEN SEND TO SAMPLE DELIVERY POSITION
C                          G.  SET FIRST CALL INDICATOR TO TRUE
C                          F.  RETURN TO MAIN
C       COMMAND DESCRIPTION: THERE ARE THREE TYPES OF COMMANDS.
C       THE "SPECIAL COMMANDS" CAUSE AN IMMEDIATE EXECUTION
C       OF SOME FUNCTION BUT DO NOT CHANGE ANY VARIABLE VALUES.
C       THE "CHANGE OF OPTION" COMMANDS SET THE VALUE OF A
C       LOGICAL VARIABLE TO .TRUE. OR .FALSE..
C       THE "CHANGE NUMERICAL ARGUMENT" COMMANDS WHEN CALLED
C       IMMEDIATELY REQUEST A NEW VALUE FOR THE VARIABLE
C       WHICH IS IMPLIED IN THE COMMAND.  THESE COMMANDS
C       SHOULD BE TYPED FOLLOWED BY A CARRIAGE RETURN.  AT THIS
C       TIME GBA MONITOR SHOULD RESPOND WITH A " = " AND
C       THE NEW VALUE OF THE VARIABLE MAY BE ENTERED.
C
C       SPECIAL COMMANDS-
C       .GO     BEGIN EXECUTION WITH PRESENT ARGUMENT VALUES.
C       .RS     RESTART INTERRUPTED TASK -NO VARIABLE INITIALIZATION
C       .HL     GIVE PRESENT STATUS OF ALL VARIABLES
C       .HLTIME GIVE HEATING TIMES
C       .HLPWR  GIVE HEATING POWERS
C       .HLOPT  GIVE STATUS PROGRAMMING OPTIONS
C       .HLSCAN GIVE SCAN VARIABLES STATUS
C       .HLCOM  GIVE LIST OF ALL GBA MONITOR COMMANDS
C       .HLLPT  GIVE STATUS OF ALL VARIABLES ON THE LINE PRINTER
C       .HLPOS  GIVE STATUS OF POSITION VARIABLES
C*S1,S2 .HLRUN  GIVE TOTAL NUMBER OF RUNS REQUIRED FOR COMPLETION
C
C       CHANGE OF OPTION COMMANDS-
C       .PLT    PLOT ON ADDS
C       .NPLT   DO NOT PLOT ON ADDS
```

```
C       .LPT    OUTPUT DATA TO LINE PRINTER
C       .NLPT   DO NOT OUTPUT DATA TO THE LINE PRINTER
C       .RAD    USE RADIATION PROGRAMMING FOR ATOMIZATION
C       .NRAD   USE POWER PROGRAMMING FOR ALL HEATING STEPS
C       .HDF    USE HYDROGEN DIFFUSION FLAME
C       .NHDF   DO NOT USE HYDROGEN DIFFUSION FLAME
C       .WAIT   WAIT AFTER EACH COMPLETE RUN FOR OPERATOR INTERVENTION
C       .NWAIT  DO NOT WAIT; GO TO NEXT RUN IMMEDIATELY
C       .SCAN   ALLOW SCANNING
C       .NSCAN  DO NOT ALLOW SCANNING
C       .SAMP   USE AUTO SAMPLER
C*L     .NSAMP  DO NOT USE AUTO SAMPLER
C
C       CHANGE OF NUMERICAL ARGUMMENT COMMANDS-
C       .DEST   CHANGE THE DESOLVATE TIME TO THE NEXT VALUE ENTERED
C       .ASHT   CHANGE THE ASH TIME TO THE NEXT VALUE ENTERED
C       .ATMT   CHANGE THE ATOMIZE TIME TO THE NEXT VALUE ENTERED
C       .DELT   CHANGE THE DELAY TIME TO THE NEXT VALUE ENTERED
C       .DESP   CHANGE THE DESOLVATE POWER TO THE NEXT VALUE ENTERED
C       .ASHP   CHANGE THE ASH POWER
C       .ATMP   CHANGE THE ATOMIZE POWER
C       .VER    CHANGE THE VERTICAL STARTING POSITION
C       .HOR    CHANGE THE HORIZONTAL STARTING POSTION
C       .VERI   CHANGE THE VERTICAL INCREMENT
C       .HORI   CHANGE THE HORIZONTAL INCREMENT
C       .VERL   CHANGE THE VERTICAL LIMIT
C       .HORL   CHANGE THE HORIZONTAL LIMIT
C       .NMRUN  CHANGE NUMBER OF RUNS PER POSITION
C       .NMSCAN CHANGE NUMBER OF SCANS
C       .TITLE  READ IN 72 CHARACHTERS OF EXPERIMENT ID
C*N     .AMNT   CHANGE THE SAMPLE SIZE
C
C       *NOTES ON ADDING COMMANDS
C
C       1.  THE COMMAND MUST BE 6 CHARACTERS OR LESS
C
C       2.  DECIDE WHAT TYPE OF COMMAND IT IS:
C               A.  SPECAIL 1- REQUIRES MORE THAN ONE EXECUTEABLE
C                   COMMAND, BUT DOES NOT CHANGE THE VALUE OF A VARIABLE
C               B.  SPECAIL 2- LIKE A. BUT ONLY ONE EXECUTEABLE COMMAND
C               C.  LOGICAL-  CHANGES A LOGIC VARIABLE'S VALUE
C               D.  NUMERICAL- CHANGES VALUE OF NUMERICAL VARIABLE
C
C       3.  INSERT PROPER CODE AT POINTS MARKED C*S1,C*S2,C*N, OR
C       C*L DEPENDING ON WHAT TYPE OF COMMAND IS TO BE INSERTED
C       USE THE EXAMPLE OF A SIMILIAR TYPE OF COMMAND TO DETERMINE THE
C       EXACT CODE TO BE INSERTED.
C
C       4.  IF A NUMERICAL OR LOGICAL COMMAND IS INSERTED YOU MUST
C       ALSO UPDATE THE COMMON,LOGICAL, AND DATA STATEMENTS OF THE
C       AFFECTED PROGRAMS SUCH AS  MAIN AND EPBKDT.F4.
C
C
C       ADDITIONAL COMMANDS VERSION *1E AND VERSION 2.01
C
C       .REWIND SET INPUT SPECIFICATION FILE BACK TO BEGINNING
C       .LOCAL  SET MODE SWITCH TO READ FROM ADDS TERMINAL
C       .BATCH  SET MODE SWITCH TO READ FROM INPUT BATCH STREAM FILE
C       .USE1   SET TO CALL USER SUBROUTINE FROM DATA TAKING PROGRAM
C       .USE2   SET TO CALL USER SUBROUTINE FROM CALCULATION SUBROUTINE
C       .NUSE1  DISREGARD USER SUBROUTINE 1
C       .NUSE2  DISREGARD USER SUBROUTINE 2
C       .OPEN1  SET TO OUTPUT INTEGRATED ABSORBANCE DATA TO FILE1
C       .OPEN2  SET TO OUTPUT PEAK SHAPE RAW DATA TO FILE2
C       .CLOSE1 WRITE END FILE IN FILE1 AND PREVENT FUTHER WRITING ATTEMPTS
```

```
C       .CLOSE2 AS ABOVE FOR FILE2
C       .HDLPT  SET TO READ FROM KEYBOARD AND OUTPUT COMMENT
C               TO LINE PRINTER
C       .HEAD1 AND HEAD2 READ FROM KEYBORARD OUTPUT TO FILE
C       .COM1 AND COM2 READ FROM KEYBOARD OUTPUT TO SPECIFIED FILE
C       .EXIT   RETURN TO KEYBOARD MONITOR ENTER FILES IN DIRECTORY
C       .DARK   TAKE 100 DARK CURRENT READINGS AND RETURN THE AVERAGE
C       .BURN   ALLOW TEMPORARY HEATING OF THE BRAID
C       .PSIN   INITIALIZE THE POSITIONER
C       .SPIN   INITIALIZE THE SAMPLER
C       .POSDEM ALLOW USER SECIFIED POSITIONER MOVEMENTS
C       .SAMPDM ALLOW USER SPECIFIED SAMPLER ACTIONS
C       .TUNE   ALLOW TUNING OF POSITION OF SAMPLER ABOVE THE BRAID
C       .PARM   OUTPUT INSTRUMENT PARAMETERS TO OPEN OUTPUT FILES
C       .FL1    ACTIVATE DATA WRITING COMMANDS TO FILE 1
C       .NFL2   INACTIVATE DATA WRITING COMMANDS TO FILE1
C       .FL2 AND .NFL2  AS ABOVE BUT FOR FILE 2
C       .RSBAT  RESTART DATA ACQUISTION WHERE INTERRUPTED AND
C               SET MODE TO REMOTE
C       .SAMPT  SET TO USE AUTOSAMPLER DELIVERIES BUT DO NOT
C               ASK FOR SAMPLER DELIVERY SPECIFICATIONS
C       .SP1    SPARE VARIABLE ONE
C       .SP2    SPARE VARIABLE TWO
C       .BELL   RINGS TERMINAL BELL REPEATEDLY UNTIL A KEY IS STRUCK
C       .MSG    WRITE MESSAGE TO TERMINAL IF IN BATCH MODE
C       .TAG    CHANGE FILE DATA IDENTIFICATION TAG OUTPUT
C
C       ADDITIONAL COMMANDS VESION 3.01:
C       .SMP    SET TO SIMPLEX OPTIMAMIZATION MODE
C       .NSMP   TO NOT USE SIMPLEX OPTIMIZATION MODE
```

```
      SUBROUTINE INPUT
      COMMON PLTT,LPT,RAD,HDF,WAIT,DEST,ASHT,ATMT,DELT,DESP
      COMMON ASHP,ATMP,SCAN,VER,HOR,VERI,HORI,VERL,HORL
      COMMON NMSCAN,NMRUN,SAMP,TAG1
      COMMON FIRST1,ISAMC,NSCAN,HPOS,VPOS,IRUN,ISAMP
      COMMON AMNT,ISAMCA,ABSINT,XI0IDK,XIDK
      COMMON FREQ,INTST,INTSP
      COMMON /A/ISAMPL(40),NRUN
      COMMON /CBLK/MODE,SUB1,SUB2,FILE1,FILE2,DARKT,PINIT,SINIT,SUB3
      COMMON /SPBLK/SP1,SP2,SP3,SP4,SP5,SP6
      COMMON /IDEB/ITER,VINIT(5,4),PM,ISXVAR(4),ITYPE,LSMP,KR,IL
      COMMON /ISIM/ IDIM,V(5,4),R(5),ALIM(3,4),IRET(5),IMV,IWRST
      LOGICAL PLTT,LPT,RAD,HDF,WAIT,SCAN,SAMP,FIRST1
      LOGICAL SUB1,SUB2,FILE1,FILE2,DARKT,PINIT,SINIT,SNT,SUB3,LSMP
      DIMENSION XID(11),USR(11),TYDATA(7)
      DIMENSION FORMT1(2),FORMT2(2)
      DATA XID/'NO TITLE',9*' '/
      DATA USR,FILL/11*' ',1*' '/
      DATA FORMT1,FORMT2/'(A2,7G13.6)','(A2,10G13.6)'/
      DATA TYDATA/'TMAX,HPOS,VPOS,XSAMP,AMNT,AMAX,ABSINT'/
      DATA XNUM,YNUM/7.,50./
      DATA VE/3.01/
      ASSIGN 205 TO ERROR
      ASSIGN 50 TO START
C     GET PRESENT DATE FROM OS/8
      CALL DATE(IM,ID,IY)
      IY=IY-1900
      W=1.
      C=0.
      Q=-1.
      FIRST1=.T.
      WRITE(0,10)VE
10    FORMAT(' GBA VERSION: ',F6.2)
      CALL CLOCK(500)
      CALL CLTIME
C     BEGIN PROGRAM EXECUTION BY ISSUEING A PROMPT
50    WRITE(0,100)
100   FORMAT(' :',$)
      CALL ANYCHR(7)
C     PICK UP COMMAND STRING -- IF IN BATCH MODE ECHO TO TERMINAL
      READ(MODE,105)ARG
      IF(MODE.EQ.4) WRITE(0,106)ARG
105   FORMAT(A6)
106   FORMAT(' ',A6)
C-    ALLOW COMMENTS IN BATCH MODE
      IF(ARG.EQ.'/') GO TO START
C*S1  INSERT LONG SPECIAL COMMANDS HERE
      IO=3
      IF (ARG.EQ.5HHLLPT)  GO TO 107
C     IF HL IS REQUESTED GIVE PRESENT STATUS ALL VARIABLES
      IF(ARG.NE.2HHL)GO TO 115
      IO=0
107   WRITE(IO,109)XID,IM,ID,IY
109   FORMAT(' TITLE:',11A6,I2,'/',I2,'/',I2,/)
      WRITE(IO,110)PLTT,LPT,RAD,HDF,WAIT,SCAN,SAMP,LSMP
110   FORMAT(' PLT',L2,'  LPT',L2,'  RAD',L2,'  HDF',L2
     1  ,'  WAIT',L2,'  SCAN',L2,'  SAMP',L2,'  SMP ',L2,/)
      WRITE(IO,1101)SUB1,SUB2,SUB3,FILE1,FILE2,SINIT,PINIT,DARKT
1101  FORMAT(' SB1',L2,'  SB2',L2,'  SB3',L2,'  FL1',L3,'  FL2',L2,
     1  '  SPIN',L2,'  PSIN',L2,'  DARK',L2,/)
      WRITE(IO,111)DELT,DEST,ASHT,ATMT
111   FORMAT(' DELT=',F7.2,'  DEST=',F7.2,'  ASHT=',F9.2
     1  ,'  ATMT=',F7.2,/)
      WRITE(IO,112)DESP,ASHP,ATMP
112   FORMAT(' DESP=',F7.3,'  ASHP=',F7.3,'  ATMP=',F9.3,/)
```

```
      WRITE(IO,113)VER,HOR,AMNT,NMRUN,FREQ
113   FORMAT(' VER=',F8.4,'  HOR=',F8.4,'  AMNT=',F9.2,
     1  '  NMRUN=',I6,'  FREQ= ',F6.1,/)
      IF (.NOT.SCAN) GO TO START
      WRITE(IO,114)VERI,HORI,VERL,HORL,NMSCAN
114   FORMAT(' VERI=',F7.4,'  HORI=',F7.4,'  VERL=',F9.4
     1  ,'  HORL=',F7.4,'  NMSCAN=',I2,/)
      GO TO START
C     IF HLRUN REQUESTED CALCULATE NUMBER OF RUNS TO COMPLETE TASK
115   IF(ARG.NE.5HHLRUN) GO TO 120
      N=1
      M=1
      L=1
      IF(HORI.NE.0.) N=HORL/HORI
      IF(VERI.NE.0.) M=VERL/VERI
      IF (NMSCAN.GT.0) L=NMSCAN
      IRUNT=NMRUN*N*M*L
      WRITE(0,117)IRUNT
117   FORMAT(' TOTAL NUMBER OF RUNS REQUIRED=',I5)
      GO TO START
C     IF COMMAND IS GO FIRST CHECK FOR LIMIT VIOLATIONS
120   IF (ARG.EQ.'RS') RETURN
      IF (ARG.NE.'RSBAT') GO TO 1202
      MODE=4
      RETURN
1202  IF (ARG.NE.2HGO) GO TO 140
      FIRST1=.T.
      RETURN
C     GIVE HELP INFORMATION IF REQUESTED
140   IF(ARG.EQ.'HLOPT') WRITE(0,110)PLTT,LPT,RAD,HDF,WAIT,SCAN,SAMP
      IF(ARG.EQ.'HLTIME') WRITE(0,111)DELT,DEST,ASHT,ATMT
      IF(ARG.EQ.'HLPWR') WRITE(0,112)DESP,ASHP,ATMP
      IF(ARG.EQ.'HLPOS') WRITE(0,113)VER,HOR,AMNT,NMRUN,FREQ
      IF(ARG.EQ.'HLSCAN')WRITE(0,114)VERI,HORI,VERL,HORL,NMSCAN
      IF(ARG.EQ.'HLCOM')WRITE(0,141)
C*S2  INSERT SHORT SPECAIL COMMANDS HERE
141   FORMAT(' GO, HL, HLTIME, HLPWR, HLOPT, HLSCAN, HLCOM, HLLPT',/,
     1  ' HLPOS, HLRUN, PLT, NPLT, LPT, NPLT, RAD, PWR',/,
     1  ' HDF, NHDF, WAIT, NWAIT, SCAN, NSCAN, SAMP, NSAMP',/,
     1  ' DEST, ASHT, ATMT, DELT, DESP, ASHP, ATMP',/,
     1  ' VER, HOR, VERI, HORI, VERL, HORL, TITLE')
      IF(ARG.EQ.'HLCOM') WRITE(0,1411)
1411  FORMAT(' NMRUN, NMSCAN, REWIND, LOCAL, BATCH, USE1, NUSE1',/,
     1  ' USE2, NUSE2, OPEN1, CLOSE1, OPEN2, CLOSE2,',/,
     1  ' HDLPT, HEAD1, HEAD2, COM1, COM2, DARK, TUNE',/,
     1  ' BURN, POSINT, SPIN, POSDEM, SAMPDM, USE3, NUSE3',/,
     1  ' FL1, NFL1, FL2, NFL2, RS, RSBAT, BELL, MSG')
C*S1,S2,L,N     INSERT NEW COMMANDS IN LISTING OF COMMANDS ABOVE
      IF(SINIT)SNT=.T.
      IF(.NOT.SINIT)SNT=.F.
C     THE FOLLOWING IF STATMENTS CHANGE LOGICAL VARIABLE VALUES
      IF (ARG.EQ.3HPLT)       PLTT=.TRUE.
      IF (ARG.EQ.4HNPLT)      PLTT=.FALSE.
      IF (ARG.EQ.3HLPT)       LPT=.TRUE.
      IF (ARG.EQ.4HNLPT)      LPT=.FALSE.
      IF (ARG.EQ.3HRAD)       RAD=.TRUE.
      IF (ARG.EQ.4HNRAD)      RAD=.FALSE.
      IF (ARG.EQ.3HHDF)       HDF=.TRUE.
      IF (ARG.EQ.4HNHDF)      HDF=.FALSE.
      IF (ARG.EQ.4HWAIT)      WAIT=.TRUE.
      IF (ARG.EQ.5HNWAIT)     WAIT=.FALSE.
      IF (ARG.EQ.4HSCAN)      SCAN=.TRUE.
      IF (ARG.EQ.5HNSCAN)     SCAN=.FALSE.
      IF(ARG.EQ.4HSAMP)       SAMP=.TRUE.
      IF(ARG.EQ.4HSAMP)       CALL SAP
```

```
          IF(ARG.EQ.5HNSAMP)        SAMP=.FALSE.
          IF (ARG.EQ.4HDARK)        CALL DARK
          IF (ARG.EQ.4HTUNE)        CALL TUNE(SNT)
          IF (ARG.EQ.6HPOSDEM)      CALL POSDEM
          IF (ARG.EQ.6HSAMPDM)      CALL SAPDEM
          IF(ARG.EQ.4HBURN)         CALL BURNDM
          IF (ARG.NE.4HPSIN)        GO TO 14115
          CALL POS(Q,C,C)
          CALL POS(W,C,C)
          CALL POS(G,C,C)
          PINIT=.T.
14115     IF (ARG.NE.4HSPIN)        GO TO 1412
          SINIT=.TRUE.
          CALL SAPINT
C*L       INSERT NEW LOGICAL COMMANDS BEFORE THIS COMMENT AND
C         IN IF STATEMENT 142
C         IF NOT SCANNING SET SCAN PARAMETERS SO THAT MAIN PROGRAM
C         RECOGNIZES THIS
1412      IF(SCAN) GO TO 142
          HORI=0.
          VERI=0.
          HORL=HOR
          VERL=VER
C*S2,L    SHORT SPECIAL COMMANDS AND LOGICAL COMMANDS
C         IF ADDED TO THE PROGRAM SHOULD ALSO BE ADDED TO THE FOLLOWING
C         LOGICAL IF STATEMENT WHICH CHECKS FOR LEGAL COMMANDS
142       IF(ARG.EQ.3HPLT.OR.ARG.EQ.4HNPLT.OR.ARG.EQ.3HLPT
     1    .OR.ARG.EQ.4HNLPT.OR.3HRAD.EQ.ARG.OR.ARG.EQ.4HNRAD.OR.
     1    ARG.EQ.3HHDF.OR.ARG.EQ.4HNHDF.OR.ARG.EQ.4HWAIT.OR.ARG
     1    .EQ.5HNWAIT.OR.ARG.EQ.4HSCAN.OR.ARG.EQ.5HNSCAN.OR.ARG.EQ.
     1    5HHLCOM.OR.5HHLPWR.EQ.ARG.OR.ARG.EQ.6HHLTIME.OR.
     1    ARG.EQ.6HHLSCAN.OR.ARG.EQ.5HHLPOS.OR.ARG.EQ.4HSAMP.OR.
     1    ARG.EQ.5HHLOPT.OR.ARG.EQ.5HNSAMP)GO TO START
          IF(ARG.EQ.4HBURN.OR.ARG.EQ.4HTUNE)GO TO START
          IF(ARG.EQ.6HPOSDEM.OR.ARG.EQ.6HSAMPDM)GO TO START
          IF(ARG.EQ.4HFILE.OR.ARG.EQ.4HDARK) GO TO START
          IF(ARG.EQ.4HSPIN.OR.ARG.EQ.4HPSIN)GO TO START
C*N       IS THIS A CHANGE NUMERICAL ARGUMENT CONNAND
C         WHEN NEW NUMERICAL ARGUMENT COMMANDS ARE ADDED THEY
C         MUST ALSO BE ADDED TO THIS IF STATEMENT
          IF(ARG.EQ.4HDESP.OR.ARG.EQ.4HASHP.OR.ARG.EQ.4HATMP.OR.
     1    ARG.EQ.4HDELT.OR.ARG.EQ.4HASHT.OR.ARG.EQ.4HATMT.OR.
     1    ARG.EQ.5HNMRUN.OR.ARG.EQ.4HDEST.OR.ARG.EQ.6HNMSCAN.OR.
     1    ARG.EQ.3HVER.OR.ARG.EQ.4HAMNT.OR.ARG.EQ.3HHOR.OR.
     1    ARG.EQ.4HVERI.OR.ARG.EQ.4HHORI.OR.ARG.EQ.4HVERL.OR.
     1    ARG.EQ.4HHORL.OR.ARG.EQ.5HTITLE.OR.ARG.EQ.4HFREQ) GO TO 145
143       IF(ARG.EQ.'SP1'.OR.ARG.EQ.'SP2'.OR.ARG.EQ.'SP3'
     1    .OR.ARG.EQ.'SP4'.OR.ARG.EQ.'TAG') GO TO 145
          GO TO 300
145       WRITE(0,150)
150       FORMAT(' =',$)
          IF(ARG.EQ.4HDEST) READ(MODE,170)DEST
          IF(ARG.EQ.4HDELT) READ(MODE,170)DELT
          IF(ARG.EQ.4HASHT) READ(MODE,170)ASHT
          IF(ARG.EQ.4HATMT) READ(MODE,170)ATMT
          IF(ARG.EQ.4HFREQ) READ(MODE,170)FREQ
          IF(ARG.EQ.4HDESP) READ(MODE,180)DESP
          IF(ARG.EQ.4HASHP) READ(MODE,180)ASHP
          IF(ARG.EQ.4HATMP) READ(MODE,180)ATMP
          IF(ARG.EQ.5HNMRUN) READ(MODE,185)NMRUN
          IF(ARG.EQ.6HNMSCAN) READ(MODE,185)NMSCAN
170       FORMAT(F6.2)
180       FORMAT(F5.3)
185       FORMAT(I4)
          IF(ARG.EQ.'SP1')  READ(MODE,180)SP1
```

```
      IF(ARG.EQ.'SP2')   READ(MODE,180)SP2
      IF(ARG.EQ.'SP3')   READ(MODE,180)SP3
      IF(ARG.EQ.'SP4')   READ(MODE,180)SP4
      IF(ARG.EQ.'TAG')   READ(MODE,105)TAG1
C     CONTINUE CHECKING FOR NUMERICAL ARGUMENT COMMANDS
      IF(ARG.EQ.3HVER)   READ(MODE,190)VER
      IF(ARG.EQ.3HHOR)   READ(MODE,190)HOR
      IF(ARG.EQ.4HVERI)  READ(MODE,190)VERI
      IF(ARG.EQ.4HHORI)  READ(MODE,190)HORI
      IF(ARG.EQ.4HVERL)  READ(MODE,190)VERL
      IF(ARG.EQ.4HHORL)  READ(MODE,190)HORL
      IF(ARG.EQ.5HTITLE)READ(MODE,310)XID
      IF(ARG.EQ.4HAMNT)  READ(MODE,170)  AMNT
190   FORMAT(F7.5)
C*N   INSERT NEW NUMERICAL COMMAND IF CHECKS HERE
      GO TO START
C-    IF ILLEGAL COMMAND RESPOND WITH ERROR MESSAGE
205   WRITE(0,210)ARG
210   FORMAT(' ?',A6)
C-    IF IN BATCH MODE SET TO LOCAL MODE IF ERROR IS DETECTED
      MODE=0
      GO TO START
C     THIS PAGE WAS ADDED TO CONTAIN VERSION 2.01 COMMANDS
C-    COMMAND USED TO WRITE A MESSAGE ON THE TERMINAL (BATCH MODE)
300   IF(ARG.NE.'MSG')  GO TO 305
      READ(MODE,310)USR
      WRITE(0,303)USR
303   FORMAT(' MSG',11A6)
C-    COMMAND TO PUT COMMENT ON THE LINE PRINTER
305   IF (ARG.NE.'HDLPT') GO TO 330
      READ(MODE,310)USR
310   FORMAT(11A6)
      WRITE(3,320)USR
320   FORMAT(1H0,11A6)
C-    COMMENTS AND HEADINGS TO FILES HANDLED HERE
330   IF(ARG.NE.'HEAD1') GO TO 350
      IF(.NOT.FILE1) GO TO 415
      READ(MODE,310)USR
      WRITE(5,340)USR
340   FORMAT('HD',11A6)
350   IF(ARG.NE.'HEAD2') GO TO 360
      IF(.NOT.FILE2) GO TO 415
      READ(MODE,310)USR
      WRITE(6,340)USR
360   IF(ARG.NE.'COM1') GO TO 380
      IF(.NOT.FILE1) GO TO 415
      READ(MODE,310)USR
      WRITE(5,370)USR
370   FORMAT('CC',11A6)
380   IF(ARG.NE.'COM2') GO TO 390
      IF(.NOT.FILE2) GO TO 415
      READ(MODE,310)USR
      WRITE(6,370)USR
390   IF(ARG.NE.'CLOSE1') GO TO 400
      IF(.NOT.FILE1) GO TO 415
      WRITE(5,391)
391   FORMAT('EF')
      END FILE 5
      FILE1=.F.
400   IF (ARG.NE.'CLOSE2') GO TO 401
      IF(.NOT.FILE2) GO TO 415
      WRITE(6,391)
      END FILE 6
      FILE2=.F.
401   IF(ARG.NE.'OPEN1') GO TO 404
```

```
      FILE1=.T.
      WRITE(5,402)FORMT1,XNUM
402   FORMAT('FO',2A6/'FN',G13.6)
      WRITE(5,403)VE,TYDATA
403   FORMAT('VE',G13.6,7A6)
      WRITE(5,4031)IM,ID,IY
4031  FORMAT('TI',3I3)
      IF(.NOT.LSMP)GO TO 404
      WRITE(5,4033)ITYPE,KR,IL,(ISXVAR(J),J=1,IDIMD
4033  FORMAT('SM','ITYPE',I4,' KRUN',I4,' ITER LMT',I4,' VAR OPT'4I4)
      WRITE(5,4034)((ALIM(I,J),I=1,3),J=1,IDIMD
4034  FORMAT('SL',3G13.6)
      DO 4035 I=1,IDIM+1
4035  WRITE(5,4036)(VINIT(I,J),J=1,IDIMD
4036  FORMAT('SI',4G13.6)
404   IF(ARG.NE.'OPEN2') GO TO 405
      FILE2=.T.
      WRITE(6,402)FORMT2,YNUM
      WRITE(6,403)VE
      WRITE(6,4031)IM,ID,IY
C     ADDITIONAL COMMANDS VERSION 3.01
405   IF(ARG.NE.'SMP') GO TO 406
      LSMP=.T.
      CALL PARIN
      GO TO START
406   IF(ARG.NE.'NSMP') GO TO 410
      LSMP=.F.
      GO TO START
410   IF (ARG.NE.'PARM') GO TO 413
      IFN=5
      IF(.NOT.(FILE1.OR.FILE2)) GO TO 415
      IF(FILE2.AND..NOT.FILE1) IFN=6
69    WRITE(IFN,70)PLTT,LPT,RAD,HDF,WAIT,SCAN,SAMP
70    FORMAT('PA',' PLTT',L2,'  LPT',L2,'  RAD',L2,'  HDF',L2,
     1' WAIT',L2,' SCAN',L2,' SAMP',L2)
      WRITE(IFN,71)SUB1,SUB2,SUB3,FILE1,FILE2,SINIT,PINIT,DARKT
71    FORMAT('PB',' SB1',L2,'  SB2',L2,'  SB3',L2,'  FL1',L2,
     1'  FL2',L2,' SINT',L2,' PINT',L2,'DARKT',L2)
      WRITE(IFN,72)DELT,DEST,ASHT,ATMT
72    FORMAT('PC',' DELT',G13.6,' DEST',G13.6,' ASHT',G13.6,
     1' ATMT',G13.6)
      WRITE(IFN,73)DESP,ASHP,ATMP,FREQ
73    FORMAT('PD',' DESP',G13.6,' ASHP',G13.6,' ATMP',G13.6,
     1' FREQ',G13.6)
      WRITE(IFN,74)VER,HOR,AMNT,NMRUN
74    FORMAT('PE','  VER',G13.6,'  HOR',G13.6,' AMNT',G13.6,
     1'NMRUN',G13.6)
      IF(SCAN)WRITE(IFN,75)VERI,HORI,VERL,HORL,NMSCAN
75    FORMAT(' VERI',G13.6,' HORI',G13.6,' VERL',G13.6,' HORL',
     1G13.6,'NSCAN',G13.6)
      IF(IFN.EQ.6) GO TO 413
      IFN=6
      IF(FILE2.AND.FILE1) GO TO 70
C-    LOGICAL TYPE COMMANDS
413   IF (ARG.EQ.'REWIND')     REWIND 4
      IF (ARG.EQ.'LOCAL')      MODE=0
      IF (ARG.EQ.'BATCH')      MODE=4
      IF (ARG.EQ.'USE1')       SUB1=.T.
      IF (ARG.EQ.'USE2')       SUB2=.T.
      IF (ARG.EQ.'USE3')       SUB3=.T.
      IF (ARG.EQ.'NUSE1')      SUB1=.F.
      IF (ARG.EQ.'NUSE2')      SUB2=.F.
      IF (ARG.EQ.'NUSE3')      SUB3=.F.
      IF (ARG.EQ.'EXIT')       CALL EXIT
      IF (ARG.EQ.'FL2')        FILE2=.T.
```

```
      IF (ARG.EQ.'NFL2')       FILE2=.F.
      IF (ARG.EQ.'NFL1')       FILE1=.F.
      IF (ARG.EQ.'FL1')        FILE1=.T.
      IF (ARG.EQ.'SAMPT')      SAMP=.T.
      IF (ARG.EQ.'BELL')       CALL BELL
C-    NOW CHECK TO SEE IF IT WAS A LEGAL COMMAND
      IF(ARG.EQ.'REWIND'.OR.ARG.EQ.'LOCAL'.OR.ARG.EQ.'BATCH'.OR.
     1 ARG.EQ.'USE1'.OR.ARG.EQ.'USE2'.OR.ARG.EQ.'NUSE1'.OR.
     1 ARG.EQ.'NUSE2'.OR.ARG.EQ.'OPEN1'.OR.ARG.EQ.'CLOSE1'.OR.
     1 ARG.EQ.'OPEN2'.OR.ARG.EQ.'CLOSE2'.OR.ARG.EQ.'HDLPT'.OR.
     1 ARG.EQ.'HEAD1'.OR.ARG.EQ.'HEAD2'.OR.ARG.EQ.'COM1'.OR.
     1 ARG.EQ.'COM2'.OR.ARG.EQ.'USE3'.OR.ARG.EQ.'NUSE3') GO TO 420
      IF(ARG.EQ.'PARM'.OR.ARG.EQ.'FL1'.OR.ARG.EQ.'NFL1'.OR.
     1 ARG.EQ.'FL2'.OR.ARG.EQ.'NFL2'.OR.ARG.EQ.'SAMPT'.OR.
     1 ARG.EQ.'BELL'.OR.ARG.EQ.'MSG') GO TO 420
      GO TO ERROR
415   WRITE(0,416)
416   FORMAT(' NO!! FILE IS NOT OPENED')
420   DO 430 IK=1,6
430   USR(IK)=FILL
      GO TO START
      END
```

```
C
C
C
C
C       PROGRAM FUNCTION:  PROGRAM SUPERVISES PARAMETER CHANGES NECESSARY
C                          FOR MULTIPLE SAMPLE RUN EXPERIMENTS
C
C       PROGRAMMER:  E. H. PALS
C
C       VERSION: *1E-1 (2.04)
C       DATE: 10/9/76
C       FILENAME: EPFLOW.F4
C
C       CALLING FORM:  CALL FLOW
C
C       EXTERNAL ROUTINES: NONE
C
C       SUMMARY:  THE PRIMARY FUNCTION OF THIS ROUTINE IS TO
C       KEEP TRACK OF WHERE THE POSITIONER, SAMPLER, ETC. ARE AT
C       AND WHAT CONDITIONS ARE REQUIRED FOR THE NEXT REQUESTED
C       SAMPLE RUN.  THE ROUTINE ALSO DETECTS THE COMPLETION OF
C       A TASK AND TAKES APPROPRIATE RE-INITIALIZATION STEPS.
C       CHANGES VERSION:*1B:
C       1.  ADD THE COMMON AND LOGICAL SPECIFICATIONS
C       CHANGE VERSION *1C:
C       1.  DEFINE ISAMPL AS LABELED COMMON ARRAY
C
C       CHANGES VERSIONS 1E AND 2.01:
C       1.  UPDATE COMMENTS
C       2.  UPDATE COMMON STATEMENT AND FIX POSITIONER CALL
C       3.  INSERT SWITCH REGISTER OPTION CHECKS.
C       4.  FIX MULTIPLE RUN WHEN NOT SAMPLING OR SCANNING
C
C       CHANGES VERSION 2.02:
C       1.  FORCE TO SET TO LOCAL MODE IF ERROR DURING BATCH STREAMING
C
C       CHANGES VERSION 2.03:
C       1.  LIMIT CHECKS MOVED HERE FROM INPUT.
C       2.  OPTION TO WAIT WHEN NOT PLOTTING ADDED.
```

```
C
C
C
C
C       METHOD:
C       1.  FIRST CALL?
C               YES-  INITIALIZE RUN COUNTER AND POSITION POINTERS.
C               NO-   GO TO 4
C       2.  SCANNING OR SAMPLING?
C               YES-  INITIALIZE SCAN PARAMETERS.
C               NO-   GO TO 4
C       3.  SAMPLING?
C               YES-  DECODE SAMPLE SPECIFICATION WORD AND IF IT
C                     IS ZERO GO TO 13 OTHERWISE GO TO 12.
C               NO-   GO TO 4
C       4.  COMPLETED REQUESTED NUMBER OF RUNS YET?
C               YES-  GO TO 5
C               NO-   GO TO 12
C       5.  NOT SAMPLING OR SCANNING?
C               YES-  GO TO TASK COMPLETE ROUTINE
C               NO-   GO TO 6
C       6.  SAMPLING?
C               YES-  GO TO 7
C               NO-   GO TO 8
C       7.  SAMPLE RUN COUNTER EQUALS REQUESTED NUMBER OF RUNS?
C               YES-  GO TO 8
C               NO-   GO TO 12
C       8.  REACHED VERTICAL LIMIT?
C               YES-  REINITIALIZE VERTICAL POSITION AND GO TO 9
C               NO-   GO TO 12
C       9.  COMPLEMENT HORIZONTAL DIRECTION
C       10.  REACHED HORIZONTAL LIMIT IN POSITIVE DIRECTION?
C               YES-  REINITIALIZE GO TO 11
C               NO-   GO TO 12
C       11.  COMPLETED REQUESTED NUMBER OF SCANS?
C               YES-  GO TO 13
C               NO-   GO TO 12
C       12.  READ SWITCH REGISTER AND EXECUTE REQUESTED FUNCTIONS.
C               SW8-  GO TO 13 (INTERUPT TASK BEFORE COMPLETION)
C               SW5-  WRITE SELECTED GENERAL PARAMETERS ON TERMINAL
C               SW6-  WRITE SELECTED SCAN VARIABLE STATUS TO TERMINAL
C               SW7-  WRITE SELECTED SAMPLING VARIABLE STATUS TO TERMINAL
C       13.  QUERRY OPERATOR TO ALLOW EXIT TO GBA MONITOR OR TO
C            RESTART DATA ACQUSITION WITH PRESENT CONDITIONS.
```

```
      SUBROUTINE FLOW
      COMMON PLTT,LPT,RAD,HDF,WAIT,DEST,ASHT,ATMT,DELT,DESP
      COMMON ASHP,ATMP,SCAN,VER,HOR,VERI,HORI,VERL,HORL
      COMMON NMSCAN,NMRUN,SAMP,SAMPSZ
      COMMON FIRST1,ISAMC,NSCAN,HPOS,VPOS,IRUN,ISAMP
      COMMON AMNT,ISAMCA,TAG,XI0IDK,XIDK
      COMMON FREQ
      COMMON /A/ISAMPL(40),NRUN,IGOTO
      COMMON /B/DATA1(100),DATA2(100),AMIN,AMAX,IDIR
      COMMON /CBLK/MODE,SUB1,SUB2,FILE1,FILE2,DARKT,PINIT,SINIT,SUB3
      LOGICAL PLTT,LPT,RAD,HDF,WAIT,SAMP,SCAN,FIRST1
      LOGICAL DARKT,SINIT,PINIT
      ASSIGN 1 TO DECODE
      ASSIGN 2 TO RETRN
      ASSIGN 3 TO EXIT
C
C     ALLOW WAITING HERE WHEN IN WAIT MODE AND NOT PLOTTING
C
      IF (.NOT.(.NOT.PLTT.AND.WAIT))GO TO 1100
1000  WRITE(0,1010)
1010  FORMAT(' CONTINUE(C) OR TO GBA MONITOR(M)? ',$)
      READ(0,11)AA
      IF(AA.EQ.'C') GO TO 1100
      IF(AA.NE.'M') GO TO 1000
      MODE = 0
      IGOTO = 1
      RETURN
C
C     DETERMINE IF INITAIL VARIABLE SETUP NECESSARY
C
1100  IF(.NOT.FIRST1) GO TO 100
      IGOTO=3
      IF(LIMIT(VERL,0.,26.01,'VERL').GT.0) IGOTO=1
      IF(LIMIT(HORL,-12.7,+12.7,'HORL').GT.0) IGOTO=1
C*N   INSERT VARIABLE LIMIT VIOLATION CHECKS HERE
      IF (PINIT.AND.DARKT.AND.(SINIT.OR..NOT.SAMP)) GO TO 121
      WRITE(0,1201)
1201  FORMAT(' NO!! EITHER DARK CURRENT, SAMPLER, OR POSITIONER'
     1 'NOT INITIALIZED',/)
C     IF NOT SCANNING SET SCAN VARIABLES TO COINCIDE WITH THIS
      IGOTO = 1
121   IF(IGOTO.EQ.1)RETURN
      IF(SCAN) GO TO 125
      HORI=0.
      VERI=0.
      HORL=HOR
      VERL=VER
C     IF REQUESTED WRITE HEADING ON LINE PRINTER OUTPUT
125   IF(LPT)WRITE(3,122)
122   FORMAT(1H0,'SAMP',2X,'AMOUNT',3X,'HPOS',5X,'VPOS'
     1 ,5X,'TMAX',3X,'ABSMAX',5X,'INTABS')
C     MAKE SURE ALL FLAGS SET BEFORE EXITING
      CALL CLOCK(500)
      CALL CLTIME
C     WAIT FOR POSITIONER TO INITAILIZE
      CALL POS(W,0.,0.)
      XNULL=0.
      HPOS=HOR
      VPOS=VER
C     SEND POSITIONER TO SAMPLE DELIVERY POSITION
      CALL POS(XNULL,XNULL,XNULL)
C     MAKE SURE RAD FLAG INITIALLY CLEAR
      CALL FLAG(-1.,0.,0.)
      NRUN=0
      C=0.
```

```
      IDIR=1
      ISAMC=0
      NSCAN=0
1     HPOS=HOR
      VPOS=VER
      IF(.NOT.(SCAN.OR.SAMP)) GO TO 2
      IF(HORL-HPOS)9,8,9
8     HORL=HORL-1.
9     IF(VERL-VPOS)16,17,16
17    VERL=VERL-1.
16    IF (.NOT.SAMP) GO TO RETRN
      ISAMCA=0
C
C     IF SAMPLE SPECIFICATION EQUALS ZERO GO TO EXIT ROUTINE
C
      ISAMC=ISAMC+1
      IXX=ISAMPL(ISAMC)
      IF (IXX)3,3,4
C
C     DECODE SAMPLE SPECIFICATION WORD
C
4     Y=FLOAT(ISAMPL(ISAMC))/100000.
      IRUN=IFIX(Y)
      ISAMP=IFIX((Y-FLOAT(IRUN))*100.)
      AMNT=((Y-FLOAT(IRUN)-(FLOAT(ISAMP)/100.))*1000.)+.005
C
C     IF NOT FIRST CALL BEGIN CHECKING RUN VARIABLES
C
C     CHECK RUN COUNTER
C
100   NRUN=NRUN+1
      IF(NMRUN-NRUN)101,101,2
101   NRUN=0
      IF(.NOT.(SAMP.OR.SCAN)) GO TO 3
      IF(.NOT.SAMP)GO TO 102
      IF(SAMP.AND.SCAN) GO TO 19
C
C     CHECK SAMPLE RUN COUNTER
C
      ISAMCA=ISAMCA+1
      IF(IRUN-ISAMCA)103,103,2
103   ISAMCA=0
102   IF(SAMP.AND..NOT.SCAN) GO TO DECODE
C
C     CHECK VERTICAL POSITION
C
19    VPOS=VPOS+VERI
      IF(VERL-VPOS)20,2,2
20    VPOS=VER
C
C     COMPLEMENT HORIZONTAL DIRECTION
C
      IDIR=-IDIR
      IF(IDIR-0)22,22,21
C
C     CHECK HORIZONTAL POSITION
C
22    HPOS=ABS(HPOS)+HORI
      IF(HORL-HPOS)23,2,2
21    HPOS=-HPOS
      GO TO 2
23    HPOS=HOR
C     CHECK NUMBER OF SCANS
      NSCAN=NSCAN+1
      IDIR=1
```

```
      IF(NMSCAN-NSCAN)110,110,2
110   NSCAN=0
      IF(.NOT.SAMP) GO TO EXIT
      GO TO DECODE
C
C     ALLOW TASK INTERUPTION OR RUN COUNTING PROGRESS REPORT
C
2     IGOTO = 3
      XPTS=ATMT*FREQ
      IF(LIMIT(DEST,0.,20.,'DEST').GT.0)IGOTO=1
      IF(LIMIT(ASHT,0.,30.,'ASHT').GT.0) IGOTO=1
      IF(LIMIT(ATMT,0.,5.0,'ATMT').GT.0) IGOTO=1
      IF(LIMIT(DELT,5.,60.,'DELT').GT.0) IGOTO=1
      IF(LIMIT(DESP,0.,10.00,'DESP').GT.0) IGOTO=1
      IF(LIMIT(ASHP,0.,10.00,'ASHP').GT.0) IGOTO=1
      IF(LIMIT(ATMP,0.,10.00,'ATMP').GT.0)  IGOTO=1
      IF(LIMIT(VER,0.,26.01,'VER').GT.0) IGOTO=1
      IF(LIMIT(HOR,-12.7,+12.7,'HOR').GT.0) IGOTO=1
      IF(LIMIT(VERI,-26.01,+26.01,'VERI').GT.0) IGOTO=1
      IF(LIMIT(HORI,-26.01,+26.01,'HORI').GT.0) IGOTO=1
      IF(LIMIT(AMNT,0.,200.,'AMNT').GT.0) IGOTO=1
      IF(LIMIT(FREQ,1.,1000.,'FREQ').GT.0)IGOTO=1
      IF(LIMIT(XPTS,1.,1000.,'NUMPTS').GT.0)IGOTO=1
      IF (IGOTO.EQ.3) GO TO 398
      MODE=0
      RETURN
398   IF (SWITCH(0.).EQ.0.) GO TO 399
      MODE=0
      GO TO 3
399   IF (SWITCH(1.).EQ.0.) GO TO 401
      WRITE(0,400) FIRST1,SCAN,SAMP,NMRUN,NRUN,HPOS,VPOS
400   FORMAT(' FIRST1 SCAN SAMP NMRUN NRUN HPOS VPOS',/
     1  ,L4,L7,L5,I5,I6,F6.2,F5.2)
401   IF(SWITCH(2.).EQ.0.) GO TO 403
      WRITE(0,402)IDIR,NMSCAN,NSCAN,VER,HOR,VERL,HORL
402   FORMAT(' IDIR NMSCAN NSCAN  VER  HOR  VERL HORL',/
     1  ,I4,2I6,3X,,4F5.2)
403   IF(SWITCH(3.).EQ.0.)GO TO 405
      WRITE(0,404)ISAMC,ISAMCA,IRUN,ISAMP,AMNT
404   FORMAT(' ISAMC ISAMCA IRUN ISAMP     AMNT',/
     1  ,I4,I6,I6,I5,9X,F5.2)
405   FIRST1=.F.
      IGOTO=3
      RETURN
C
C     ALLOW EXIT TO GBA MONITOR
C
3     WRITE(0,15)
15    FORMAT(' EXIT TO GBA MONITOR(M) OR DATA TAKING(D)')
      READ(MODE,11)AA
11    FORMAT(A1)
      IF(AA.EQ.'M')IGOTO=1
      IF(AA.EQ.'D')IGOTO=3
      FIRST1=.T.
      RETURN
      END
```

```
C
C
C       COMBINED ROUTINES TO ACQUIRE DATA AND CALCULATE RESULTS
C
C
C       FUNCTION:  THIS ROUTINE HANDLES DATA ACQUISTION
C
C       PROGRAMMER:  E. H. PALS
C
C       FILNAME:  EPDATA.F4
C       DATE: 9/10/76  (4/11/77)
C       VERSION:  *1B  (2.10)-T
C
C       CALLING FORM:  CALL DATA
C
C       PURPOSE:  THIS ROUTINE CONTROLS THE TAKING AND STORING
C       OF RAW DATA FROM ONE SAMPLE RUN AND CALCULATES
C       INTEGRATED ABSORBANCE.
C
C       CHANGES VERSION *1B:
C       1.  FLAG ROUTINE CALLED TO SET RAD FLAG, EPCURT NO LONGER
C       A SUBROUTINE.
C
C       CHANGES VERSION 2.01:
C       1.  DESOLVATE BEFORE TAKING 100%T DURING POSITIONER MOVE
C       2.  ADD OPTION TO CALL USER SUBROUTINE WHEN DATA TAKEN
C       3.  USE FAST A/D ACQUISTION ROUTINES
C
C       CHANGES VERSION 2.02:
C       1.  FORMAT OF FILE WRITING CHANGED
C       CHANGES VERSION 2.10:
C       1.  DATA FILE OUTPUT FORMAT IS CHANGED TO ALLOW USER
C           DEFINITION OF DATA TAG ALSO OUTPUT ORDER IS ALTERRED.
C       CHANGES VERSION T:
C       1.  MODIFIED TO ALLOW BURN OFF OF SAMPLE BETWEEN SAMPLE
C           AND BACKGROUND RUNS.
C
C       EXTERNAL ROUTINES REQUIRED:
C       1.  FREQS(FREQ),SET(ARRAY,NCON,IERR),GO
C                                  FILE-EPADCR.RA AND EPAD8M.RA
C               FREQS- STARTS THE ADC CLOCK AT REQUESTED FREQUECY
C               SET- PASSES REQUESTED NUMBER OF CONVERSIONS DATA ARRAY
C               AND IERR INFORMS USER OF ILLEGAL CALLS
C               GO- STARTS THE DATA TAKING PROCESS SET UP BY FREQS AND GO
C       2.  DAC(VOLT)                      FILE- EPDAC.RA
C               OUTPUTS VOLT VOLTS AT REFERENCE
C               INPUT OF GBA POWER SUPPLY.
C       3.  POS(I,HPOS,VPOS)               FILE- EPPOS.RA
C               I=-1; HARDWARE RESET OF POSITIONER
C               I= 1; WAIT FOR POSITIONER TO REACH DESTINATION
C               I= 0; START MOVE TO POSITION HPOS,VPOS.
C       4.  SAMP(DATA1,DATA2)              FILE- EPSAMS.RA
C               DELIVER AMNT MICROLITERS OF SAMPLE
C               SOLUTION NUMBER ISAMP.  IF ISAMP<0
C               WAIT FOR COMPLETION BEFORE RETURNING.
C       5.  CLOCK ROUTINES                 FILE- CLOCK.RA
C               CLOCK(FREQ)-  START CLOCK AT FREQ FREQUENCY
C               CLTIME-  CLEAR CLOCK COUNTER
C               TIME(SEC)-  RETURN TIME IN SEC SINCE CLTIME
C                       WAS LAST CALLED.
```

```
C
C       METHOD:
C
C       1.  WAIT FOR POSITIONER TO REACH SAMPLE DELIVERY POSITION
C       2.  WAIT FOR DELAY TIME TO ELAPSE
C       3.  IF REQUESTED DEPOSIT SAMPLE
C       4.  LATCH DESOLVATE POWER TO DAC AND WAIT FOR
C           DESOLVATE TIME TO ELAPSE
C       5.  TAKE 100 PTS OF 100%T AND CALCULATE AVERAGE
C       6.  ASH AT ASH POWER FOR REQUESTED ASH TIME
C       7.  SET UP GBA ADC CLOCK TO REQUESTED FREQUENCY, POINT RESULTS TO
C           ARRAY XVAL (DONE BY CALLING FREQS)
C       8.  SET RADIATION PROGRAMING FLAG IF REQUESTED
C       9.  LATCH TO DAC THE ATOMIZATION POWER AND WAIT FOR ATOMIZATION
C           TIME TO ELAPSE
C       10.  CLEAR RADIATION PROGRAMMING FLAG, SET DAC TO 0 VOLTS
C       11.  WAIT FOR DELAY TIME TO ELAPSE
C       12.  RE-EXECUTE STEPS 4,6-10 BUT DEPOSIT RESULTS IN BACK NOT XVAL
C       13.  CLEAR CLOCK TO 0 SECONDS (I.E. START DELAY TIME)
C       14.  START POSITIONER TOWARD SAMPLE DELIVER POSITION.
C       15.  ALLOW EXECUTION OF A USER SUBROUTINE AND IF ISKP SET=0 IN THE
C            SUBROUTINE GO TO STEP 19, OTHERWISE CONTINUE (THIS ALLOWS AN
C            ALTERNATE CALCULATION ROUTINE)
C       16.  INITIALIZE VARIABLES FOR CALCULATIONS
C       17.  CALCULATE THE INTEGRATED ABSORBANCE
C       18.  FIND ABSORBANCE MAXIMUM,MINIMUM AMD TIME OF MAXIMUM ABS
C       19.  ALLOW EXECUTION OF A USER DEFINED SUBROUTINE.  IF ISKP IS SET
C            IN THE SUBROUTINE TO 0 EXIT TO MAIN, IF ISKP = 1 GO TO 22
C            IF ISKP NOT 0 OR 1 CONTINUE TO 20
C       20.  WRITE RESULTS ON LINE PRINTER IF REQUESTED
C       21.  WRITE RESULTS ON EITHER OF TWO FILES
C       22.  PLOT REQUESTED?
C               NO-  RETURN TO MAIN
C               YES- PACK A REPRESENTATIVE SET OF TRANSMITTANCE
C                    VALUES INTO DATA ARRAYS FOR FUTURE PLOTTING
C       23.  RETURN TO MAIN
```

```
      SUBROUTINE DATA
      COMMON PLTT,LPT,RAD,HDF,WAIT,DEST,ASHT,ATMT,DELT,DESP
      COMMON ASHP,ATMP,SCAN,VER,HOR,VERI,HORI,VERL,HORL
      COMMON NMSCAN,NMRUN,SAMP,TAG1
      COMMON FIRST1,ISAMC,NSCAN,HPOS,VPOS,IRUN,ISAMP
      COMMON AMNT,ISAMCA,ABSINT,XI0IDK,XIDK
      COMMON FREQ
      COMMON /B/DATA1(100),DATA2(100),AMIN,AMAX,IDIR
      COMMON /CBLK/MODE1,SUB1,SUB2,FILE1,FILE2,DARKT,PINIT,SINIT,SUB3
      COMMON /SPBLK/BRNT,BRNP
      LOGICAL PLTT,LPT,RAD,HDF,WAIT,SCAN,SAMP,FIRST1
      LOGICAL F,SUB1,SUB2,FILE1,FILE2
      DIMENSION BACK(1000),XVAL(1000)
      DATA ISKP/2/
C
C     WAIT FOR POSITIONER TO REACH DESTINATION
C
      XNULL=0.
      WAITP=1.
      CALL POS(WAITP,0.,0.)
C
C     WAIT FOR DELAY TIME TO ELAPSE
C
10    CALL TIME(XT)
      IF (XT.LT.DELT) GO TO 10
C
C     DEPOSIT SAMPLE IF REQUESTED
C
      IF(.NOT.SAMP) GO TO 90
      XSAMP=FLOAT(ISAMP)
      AMNTT=AMNT
      CALL DAC(DESP)
      CALL SAMPD(XSAMP,AMNTT)
      CALL DAC(XNULL)
C
C     DESOLVATE BEFORE TAKING 100%T
C
      CALL CLOCK(500)
90    CALL CLTIME
      CALL DAC(DESP)
15    CALL TIME(XT)
      IF(XT.LT.DEST) GO TO 15
      CALL CLTIME
      CALL DAC(0.)
C
C     GET I(0)+I(DARK), AVERAGE 100 PTS AND RETURN
C
      CALL POS(XNULL,HPOS,VPOS)
      CALL POS(WAITP,HPOS,VPOS)
      FFREQ=100.
      CALL FREQS(FFREQ)
      NCON=100
      CALL SET(BACK,NCON,IERR)
      CALL GO
      SUM=0.
      DO 91 I=1,100
91    SUM=SUM+BACK(I)
      XI0IDK=SUM/100.
      NPTS=IFIX(ATMT*FREQ)
C     START HEATING STEPS
      F=.T.
      DO 60 J=1,2
23    CALL CLTIME
      CALL DAC(ASHP)
25    CALL TIME(XT)
```

```
      IF (XT.LT.ASHT) GO TO 25
C     START ACQUIRING DATA
      SETF=1.
      CLEAR=-1.
      FQ=FREQ
      CALL FREQS(FQ)
      IF(F) CALL SET(XVAL,NPTS,IERR)
      IF(.NOT.F) CALL SET(BACK,NPTS,IERR)
40    IF(RAD) CALL FLAG(SETF,0.,0.)
      CALL DAC(ATMP)
      CALL GO
      F=.F.
      CALL FLAG(CLEAR,CLEAR,0.)
      XNULL=0.
C     WAIT FOR 10 SECOND DELAY
      CALL DAC(XNULL)
      IF (J.EQ.2) GO TO 65
      CALL CLTIME
      CALL FLAG(CLEAR,CLEAR,CLEAR)
      CALL DAC(BRNP)
49    CALL TIME(XT)
      IF(XT.LT.BRNT) GO TO 49
      CALL DAC(XNULL)
      CALL CLTIME
50    CALL TIME(XT)
      IF (XT.LT.3.) GO TO 50
60    CONTINUE
C     START DELAY CLOCK
65    CALL CLTIME
C     START POSITIONER BACK TO ORIGIN
      CALL POS(XNULL,.01,XNULL)
C
C
C
C     THIS PORTION OF THE ROUTINE WAS ORIGINALLY AN INDEPENDENT
C     CALCULATION SUBROUTINE.  CONSULT EPCALC.F4 VERSION 1
C     FOR ORGINAL VERSION OF THIS ROUTINE.
C
C
C
C
C     ALLOW INTERPOSISITION OF A USER SUBROUTINE FOR MASSAGING
C     THE RAW DATA OR FOR ALTERNATE CALCULATION METHODS
C
      IF (SUB1) CALL SUBONE(ISKP,XVAL,BACK)
      IF (ISKP.EQ.0) GO TO 105
C
C     SET UP VARIABLES FOR ABSORBANCE CALCULATIONS
C
      ABSINT=1.
      AMIN=10.**10
      AMAX=-AMIN
      Q=XI0IDK-XIDK
      TMAX=0.
C
C-    NOW CALCULATE THE INTEGRATED ABSORBANCE
C
      DO 100 I=1,NPTS
      TDIF=Q/(XVAL(I)-BACK(I)+Q)
      IF (AMAX.LT.TDIF) AMAX=TDIF
      IF (AMIN.GT.TDIF) AMIN=TDIF
      IF (AMAX.EQ.TDIF) TMAX=FLOAT(I)/FREQ
100   ABSINT= ABSINT*TDIF
      ABSINT=ABS(ABSINT)
      ABSINT=ALOG10(ABSINT)
```

```
      AMAX=ABS(AMAX)
      AMAX=ALOG10(AMAX)
      AMIN=ABS(AMIN)
      AMIN=ALOG10(AMIN)
C
C-    ALLOW USER SUBROUTINE FOR DATA TREATMENT AND/OR ANALYSIS
C
105   ISKP=2
      IF(SUB2) CALL SUBTWO(ISKP,XVAL,BACK)
      IF (ISKP.EQ.0)RETURN
      IF (ISKP.EQ.1) GO TO 135
C
C-    IF REQUESTED OUTPUT THIS DATA TO LINE PRINTER
C
      IF(.NOT.LPT) GO TO 120
      WRITE(3,110)ISAMP,AMNT,HPOS,VPOS,TMAX,AMAX,ABSINT
110   FORMAT(' ',I3,2X,F8.2,1X,F7.3,2X,F7.3,1X,F7.2,F8.2,2X,F10.2)
C
C-    IF REQUESTED WRITE DATA ON OUTPUT FILE
C
120   IF(.NOT.FILE1)GO TO 135
      WRITE(5,125)TAG1,TMAX,HPOS,VPOS,XSAMP,AMNT,AMAX,ABSINT
125   FORMAT(A2,7G13.6,/)
C
C     IF PLOT REQUESTED SET UP DATA ARRAYS IN COMMON TO PASS DATA
C
135   IF (.NOT.PLTT.AND.(SWITCH(4.).NE.1.))GO TO 145
      FINC=(FLOAT(NPTS))/100.
      PT=1.
      DATA1(1)=XVAL(1)
      DATA2(1)=BACK(1)
      DO 140 I=2,100
      PT=PT+FINC
      IPT=FLOAT(PT)
      DATA1(I)=XVAL(IPT)
140   DATA2(I)=BACK(IPT)
C
C     ALLOW OUTPUTTING SHAPE TO FILE STRUCTURED DEVICE
C
145   IF(.NOT.FILE2) RETURN
      DO 146 I=1,NPTS
146   XVAL(I)=ALOG10(ABS(Q/(XVAL(I)-BACK(I)+Q)))
      DO 147 I=1,NPTS,10
147   WRITE(6,148) TAG1,(XVAL(K),K=NPTS,NPTS+10)
148   FORMAT(A2,10G13.6)
      WRITE(6,151)
151   FORMAT('ER')
      RETURN
      END
```

```
C
C
C       FUNCTION:  PLOTTER ROUTINE FOR GBA SAMPLE RUN
C
C       PROGRAMMER:  E. H. PALS
C
C       VERSION:  *1B-1          (2.02)
C       DATE:  9/20/76           (12/31/76)
C       FILE:  EPPLOT.F4         (EPPTV2.F4)
C
C       CALLING FORM: CALL POT
C
C       SUMMARY:  THIS ROUTINE TAKES THE RAW DATA COLLECTED
C       IN DATA1 AND DATA2, CALCULATES ABSORBANCES AND THEN
C       PLOTS THE RESULTS ON THE ADDS.  IF THE WAIT OPTION
C       WAS REQUESTED, OPTIONS ARE AVAILABLE TO GET A
C       HARDCOPY PLOT ON THE LPT, OR A PLOT OF
C       THE TRANSMITTANCE OF THE BACKGROUND RUN.
C
C       EXTERNAL ROUTINES:  THIS ROUTINE USES ALL THE
C       ROUTINES IN ERJ'S ADDS PLOTTING PACKAGE.
C       CHANGES VERSION *1B:
C       1.  TAKE DATA OUT OF COMMON
C       CHANGES VERSION 2.01:
C       1.  RESTRICT PLOTTING OPERATION TO THE 100 PTS IN
C           ARRAYS DATA1 AND DATA2.
C
C
C
C       METHOD:
C
C       1.  CALL PLINT TO SET UP PLOTTING LIMITS
C       2.  CALL AXIS TO DRAW AXIS
C       3.  CALL LABEL TO LABEL AXIS TIME AND ABSORBANCE
C       4.  CALCULATE INDIVIDUAL ABSORBANCES FOR DATA STORED
C           IN PASSED ARRAYS
C       5.  CALL PLT TO PLOT EACH POINT AS IT IS CALCULATED
C       6.  WAIT REQUESTED?
C               NO-  RETURN TO MAIN
C               YES- A. PRINT HARDCOPY IF REQUESTED
C                    B. IS PLOT OF BACKGROUND REQUESTED?
C                       NO-  RETURN TO MAIN
C                       YES- GO TO 7
C       7.  FIND MAXIMUM AND MINIMUM TRANMITTANCES
C       8.  PASS MAX ANS MIN, DRAW AXIS AND LABEL
C       9.  CALCULATE AND PLOT INDIVIDUAL TRANSMITTANCES
C       10.  WAIT FOR USER TO REQUEST CONTINUING
```

```
            SUBROUTINE POT
      COMMON PLTT,LPT,RAD,HDF,WAIT,DEST,ASHT,ATMT,DELT,DESP
      COMMON ASHP,ATMP,SCAN,VER,HOR,VERI,HORI,VERL,HORL
      COMMON NMSCAN,NMRUN,SAMP,SAMPSZ
      COMMON FIRST1,ISAMC,NSCAN,HPOS,VPOS,IRUN,ISAMP
      COMMON AMNT,ISAMCA,ABSINT,XI0IDK,XIDK
      COMMON FREQ
      COMMON /B/DATA1(100),DATA2(100),AMIN,AMAX,IDIR
      DIMENSION PLTDAT(224),XTXT(1),YTXT(2),YYTXT(2)
      LOGICAL PLTT,LPT,RAD,HDF,WAIT,SCAN,SAMP,FIRST1
      DATA XTXT/' TIME '/, YTXT/'ABSORBANCE'/
      DATA YYTXT/'BKG %T'/
C-    SET UP PLOTTING LIMITS INTIALIZE
      AATMT=ATMT
      AAMIN=AMIN
      AAMAX=AMAX
      CALL PLINT(0.,AATMT,AAMIN,AAMAX,PLTDAT)
C-    CREATE AXIS
      CALL AXIS(1,1,1)
C-    LABEL AXIS
      CALL LABEL(XTXT,6,YTXT,10)
      Q=XI0IDK-XIDK
      NPTS=IFIX(ATMT*FREQ)
      DO 10 I=1,100
      X=ATMT*FLOAT(I)/100.
      Y=ALOG10(ABS(Q/(DATA1(I)-DATA2(I)+Q)))
10    CALL PLT(-1,X,Y,PLTDAT)
      WRITE(0,20)AMIN,AMAX,ABSINT
20    FORMAT('   AMIN',G13.6,'   AMAX',G13.6,'   ABSINT',G13.6)
C
C     IF WAIT IS NOT REQUESTED RETURN IMMEDIATELY
C
35    IF (.NOT.WAIT) RETURN
70    WRITE(0,80)
80    FORMAT(' BACKGROUND ADDS PLOT? ',$)
      READ(0,50) ARG
50    FORMAT(A1)
      IF(ARG.EQ.'N') GO TO 100
      IF(ARG.NE.'Y') GO TO 70
C     THIS PORTION OF THE ROUTINE PLOTS THE BKG
C     NOTE THAT 0 T=XIDK AND 100T=XI0IDK-XIDK
      TRMAX=-10.**10
      TRMIN=-TRMAX
      DO 85 I=1,NPTS
      TT=DATA2(I)
      IF (TRMAX.LT.TT) TRMAX=TT
85    IF (TRMIN.GT.TT) TRMIN=TT
      TRMAX=((TRMAX-XI0IDK)/Q)*100.
      TRMIN=((TRMIN-XI0IDK)/Q)*100.
      CALL PLINT(0.,ATMT,TRMIN,TRMAX,PLTDAT)
      CALL AXIS(1,1,1)
      CALL LABEL(XTXT,6,YYTXT,6)
      DO 90 I=1,100
      T=((DATA2(I)-XI0IDK)/Q)*100.
      X=ATMT*FLOAT(I)/100.
90    CALL PLT(-1,X,T,PLTDAT)
100   WRITE(0,110)
110   FORMAT(' CONTINUE? TYPE Y',$)
      READ(0,50)ARG
      IF(ARG.NE.'Y') GO TO 100
      RETURN
      END
```

```
/
/
/
/       PROGRAM FUNCTION: 8 CODE FOR FAST ADC TO F4 CONVERSIONS
/
/       PROGRAMMER: E. H. PALS
/
/       VERSION:  *1C   (2.01)
/       DATE:  11/15/76 (12/7/76)
/       FILENAME:  EPAD8M.RA
/
/       EXTERNAL ROUTINES:  THIS ROUTINE IS USED IN CONJUNCTION WITH
/       THE RALF PORTION OF THE FAST COVERSION PACKAGE (EPADRA.RA).
/
/       LOADING INSTRUCTIONS:  MUST BE LOADED IN LEVEL 0.
/
/       ERROR CONDITIONS:  IF THE REQUESTED CONVERSION FREQUENCY
/       IS TO FAST FOR DATA STORAGE BY THIS ROUTINE AN ERROR *HALT*
/       WITH THE ACCUMULATOR =1 WILL RESULT.  THIS
/       SHOULD NEVER OCCUR FOR A FREQUENCY REQUESTED OF 1KHZ OR
/       LESS.  HIGHER FREQUENCIES ARE POSSIBLE IF THE NUMBER OF
/       DATA POINTS REQUESTED IS SMALL.
/
/       METHOD:  THIS ROUTINE IS THE COMBINED 8MODE PIPORTIONS
/       OF THE INDIVIDUAL ROUTINES EPFREQ.RA AND EPADC8.RA.  NOTES
/       CONCERNING EACH OF THESE PORGRAMS ARE LISTED TOGETHER
/       HERE FOR CONVENIENCE.
/
/       CHANGES VERSION *1B EPADC8.RA:
/       1.  EAE INSTRUCTION SCA=7441 NOT 7411.
/       2.  CORRECT DATA FIELD COUNTER DO NOT NEGATE POINTER.
/       CHANGES VERSION *1C:
/       1.  REWRITE SUBROUTINE CALL TO CORRECT PROBLEM IN
/       CROSSING DATA FIELD BOUNDRIES.
/
/       CHANGES VERSION *1B EPFREQ.RA(8MODE CODE):
/       1.  BECAUSE OF THE NEW MODE OF DATA STORAGE IT IS NOT NECESSARY
/       TO OPEN SYSTEM PAGE ZERO TO HOLD A CROSS PROGRAM
/       COMMUNICATION FLAG.
/
/       CHANGES VERSION 2.01:
/       1.  THE BRAID POWER IS SET TO ZERO BEFORE EXITING.
/
/       NOTE:  THIS PROGRAM IS WRITTEN FOR AN A/D CONVERTOR THAT
/       IS WIRED FOR A STRAIGHT BINARY OUTPUT. THE NUMBERS RETURNED
/       ARE POSITIVE FLOATING VARIABLES BETWEEN 0. AND 4096.  THE
/       EXTENDED ARITHMETIC ELEMENT IS USED TO AID IN THE CONVERSION
/       TO A FORTRAN 4 VARIABLE COMPATIBLE FORM.  BRIEFLY, THE AC
/       AND MQ ARE CLEARED THEN THE A/D VALUE LOADED INTO THE AC
/       THE AC AND MQ ARE THEN SWAPPED AND NORMALIZED.  THE STEP COUNTER
/       IS USED TO FORM THE F4 EXPONET AND THE AC IS USED
/       TO FORM THE FIRST WORD OF THE F4 VARIBLE MANTISSA.
/
/
/
////////////////////////////////////////////////
/       ADC8 IS ALL 8 MODE CODE                /
/       THIS PORTION PICKS UP VALUES SENDS TO F4/
////////////////////////////////////////////////
/
/       DEFINE I/O INSTRUCTIONS AND EAE INSTRUCTIONS
/
        SCA=7441        /EAE STEP COUNTER OR'D INTO ACC
        CAM=7621        /CLEAR AC AND MQ
        SWP=7521        /SWAP THE AC AND MQ
```

```
        NMI=7411        /NORMALIZE TO F4 LIKE VARIABLE
        SKPAD=6532      /SKIP ON A/D FLAG
        READB=6534      /READ A/D BUFFER
        CLRFG=6524      /CLEAR A/D FLAG
/
        FIELD1  ADC8
        0               /SAVE RETURN
        IOF             /DON'T BOTHER ME I'M BUSY
        CAM             /CLEAR AC AND MQ
        TAD AR          /PICK UP DATA FIELD
        AND N7          /MASK
        CLL RTL         /MOVE TO
        RAL             /PROPER BITS
        TAD N6201       /SKELETON OF CDF INSTR
        DCA .+1         /SET TO EXECUTE AS
        0               /IN LINE CODE
        TAD .-1         /SAVE FOR FUTURE REFERENCE
        DCA CCDF        /ELSEWHERE
        JMP CHECK       /DON'T CHECK IF TO SLOW ON ENTERING
/
/
/
////    THE FOLLOWING CODE IS RE-EXECUTED UNTIL ALL DESIRED VALUES
////    HAVE BEEN OBTAINED.
/
/
AGAIN,  SKPAD           /GOING TO SLOW?
        JMP CHECK       /NO, CONTINUE
        CLA             /CLEAR AC FOR POWER LEVEL =0
        6111            /CLEAR BRAID POWER SUPPLY LEVEL TO OFF
        IAC             /YES, LOAD 1 INTO ACC
        HLT             /***ERROR HALT WITH AC=1***
CHECK,  SKPAD           /GOT A VALUE?
        JMP .-1         /NO, WAIT
        CAM             /CLEAR AC AND MQ
        READB           /YES, READ IT
        CLRFG           /AND CLEAR THE FLAG
        SWP             /SWAP AC AND MQ SO EAE WONT THINK NEG
        NMI             /USE EAE TO NORMALIZE
        DCA TEMP        /STORE TEMPORARILY
        SCA             /PICK UP NUM OF STEPS
        CIA             /NEGATE
        TAD N27         /ADD 27(OCTAL)
        DCA% AR+1       /DEPOSIT EXPONET
        ISZ AR+1        /INC POINTER, END DF?
        SKP             /NO, CONTINUE
        JMS SUB         /YES, GO CHANGE THE DATA FIELD
        TAD TEMP        /GET MANTISSA
        DCA% AR+1       /DEPOSIT IN F4 MANTISSA WORD1
        ISZ AR+1
        SKP
        JMS SUB
        DCA% AR+1       /DEPOSIT 0 IN 2 WORD F4 MANTISSA
        ISZ AR+1
        SKP
        JMS SUB
        ISZ INUM        /DONE YET?
        JMP AGAIN       /NO, GET ANOTHER POINT
        ION             /ALL RIGHT I'M DONE
        CDF CIF 0       /YES, GET SET TO RETURN
        JMP% ADC8
/
/
/
////    THIS CODE CHANGES THE DATA FIELD
```

```
/
SUB,    0               /SAVE RETURN LOCATION
        CLA CLL
        TAD N10         /TO INCREMENT CHANGE DATA FIELD INSTRUCTION
        TAD CCDF        /PICK UP LAST CHANGE DATA FIELD INSTRUCTION
        DCA .+1         /SET TO EXECUTE AS IN LINE CODE
CCDF,   0               /DO IT
        CLA CLL
        JMP% SUB        /RETURN POINTING TO NEXT DATA FIELD
        ENTRY   AR
AR,     0000            /WILL CONTAIN DF IN BITS 9-11
        0000            /WILL CONTAIN THE ADDRESS
N7,     0007            /FOR MASKING DATA FIELD
N10,    0010            /TO INCREMENT DATA FIELD
N6201,  6201            /SKELETON OF CDF INSTR.
TEMP,   0000            /TEMPORARY MANTISSA STORAGE
N27,    0027            /NEEDED TO CORRECT EXPONET
        ENTRY   MENUMB
MENUMB, 0000            /ROOM FOR PASSING NUMBER OF CONVERSIONS
        0000
MNUM,   0000            /8MODE VARIABLE IS HERE
/
////////////////////////////////////////////////////////
/       8 MODE CODE FOR SETTING ADC HARDWARE CLOCK      /
////////////////////////////////////////////////////////
/
/       BIT PATTERNS FOR THE RATE ENABLE WORD
/       RATE=(M)*(10EY)
/       AC6 MUST BE SET TO ENABLE CLOCK CHIP
/
/       AC7     0    0    1    1
/       AC8     0    1    0    1
/       -------------------------
/       M=      0    1    2    5
/
/       AC9     1    0    0    0    0    1    1    1
/       AC10    0    1    1    0    0    1    1    0
/       AC11    0    1    0    1    0    1    0    1
/       ---------------------------------------------
/       Y=      4    3    2    1    0   -1   -2   -3
/       I/O DEFINITIONS
        RATE=6521       /LATCH RATE WORD AND START IF AC6=0
        CLRFG=6524      /CLEAR A/D FLAG
        ENTRY   #FREQ
#FREQ,  0               /SAVE FOR RETURN
        IOF             /CRITICAL CODE HERE
        CLRFG           /MAKE SURE FLAG INITIALLY CLEAR
        CLA CLL
        TAD M8+2        /PICK UP M
        RTL
        RAL             /PUT IN PROPER POSITION
        TAD E8          /PICK UP CLOCK ENABLE BIT
        TAD EXP8+2      /PICK UP EXPONET
        RATE            /LATCH OUT AND START CLOCK
        CLA CLL         /SET AC TO 7777
        ION             /ENABLE INTERRUPT SYSTEM
        CDF CIF 0       /RETURN
        JMP% #FREQ
        /DEFINE 8 MODE CONSTANTS
        ENTRY   EXP8
EXP8,   0;0;0;          /EXPONENT HERE
        ENTRY   M8
M8,     0;0;0           /MANTISSA HERE
E8,     0040            /CLOCK ENABLE BIT
```

```
/       PROGRAM FUNCTION:  THIS MODULE WITH THE 8MODE MODULE EPAD8M.RA
/       CONTROL THE GBA A/D CONVERTOR.
/
/       PROGRAMMER:  E. H. PALS
/
/       VERSION: *1A
/       DATE: 12/7/76
/       FILENAME:  EPADCR.RA
/
/       LOADING INSTRUCTIONS:  MAY BE LOADED IN ANY LEVEL.
/       EXTERNAL ROUTINES: NONE OTHER THAN F4 LIBRARY
/       ROUTINES AND THOSE IN THE 8MODE PACKAGE (EPAD8M.RA).
/
/       METHOD:  THIS MODULE IS A PACKAGE OF THREE SUBROUTINES-
/       CALLING FORMS:  CALL FREQS(FFREQ)
/                       CALL SET(ARRAY,NUM,IERR)
/                       CALL GO
/       VARIABLE DEFINITIONS:   FREQ-CONVERSION FREQUENCY
/                               ARRAY- NAME OF DATA STORAGE ARRAY
/                               NUM- NUMBER OF CONVERSIONS REQUESTED
/                               IERR- RETURN VALUE >0 INDICATES ERROR
/                               FOR FURTHER EXPLANATIONS CHECK INDIVIDUAL
/                               PROGRAM LISTINGS THAT FOLLOW.
/       NOTE:  SUBROUTINES GO AND FREQ WERE ENTIRELY COMPOSED
/       BY THE AUTHOR.  SET IS AN EDITED VERSION OF THE RALF LISTING
/       FILE PRODUCED BY COMPILING A FORTRAN 4 SOURCE FILE.  THIS
/       SOURCE FILE MAY BE FOUND ELSEWHERE IN THE GBA DOCUMENTATION.
/
/       NOTE:  THE 8MODE CODE OF THIS PACKAGE COULD HANDLE
/       UP TO 4096 REQUESTED CONVERSIONS AND SET COULD EASILY BE
/       MODIFIED TO ACCOMDATE THIS HOWEVER THE LIMIT HAS BEEN ARBITRARILY
/       SET AT 2000 CONVERSIONS MAXIMUM.
/
/       ERROR CONDITIONS:  IF NUM IS <0 IERR IS RETURNED AS 1, IF NUM>2000
/       IERR IS RETURN AS 2.  IERR IS RETURNED AS 0 WHEN THERE ARE
/       NO ERRORS.  AN ERROR *HALT* OCCURS WITH 1 IN THE ACCUMULATOR
/       IF THE REQUESTED CONVERSIONS FREQUENCY IS TO FAST FOR PROGRAM
/       EXECUTION.
/
/       PROGRAM FUNCTION: START GBA ADC AT REQUESTED
/       FREQUENCY
/
/       PROGRAMMER:  E.H.PALS
/
/       VERSION:  *1B
/       DATE:  10/5/76
/       FILENAME:  EPFREQ.RA
/
/       CALLING FROM:  CALL FREQS(FFREQ)
/         WHERE - FFREQ IS FLOATING VALUE OF
/                 FREQUENCY CONVERSION REQUESTED.
/
/       NOTE:  THE FREQUENCY OF CONVERSION REQUESTED IS ROUNDED DOWN
/       TO MULTIPLES OF 1., 2., AND 5.
/
/       EXTERNAL ROUTINES:  NONE, BUT THIS ROUTINE
/       IS COMMONLY USED WITH SET AND GO TO PICK UP VALUES
/       FROM THE ADC.
/
        EXTERN  EXP8    /IN EXTERNAL 8MODE MODULE
        EXTERN  M8      /THIS ONE ALSO
        EXTERN  *FREQ   /NAME OF 8MODE ENTRY POINT
        SECT    FREQS
        JA      *FRQST
BPFREQ, F 0.
```

```
XRFREQ,  F 0.
FRQTMP,  F 0.
TEMP,    F 0.0
EXP,     F 0.
         0
FRQRTN,  JA       .
         BASE     0
#FRQST,  STARTD
         FLDA     10*3
         FSTA     FRQRTN,0
         FLDA     0
         SETX     XRFREQ
         SETB     BPFREQ
         BASE     BPFREQ
         FSTA     BPFREQ
         LDX      1,1
         FLDA%    BPFREQ,1
         FSTA     FRQTMP
         STARTF
/
/        RALF CODE CALCULATE VALUES FOR SETTING RATE
/
         FCLA
         FSTA     EXP              /SET EXP = 0
         FLDA%    FRQTMP           /LOAD FREQ
         FSTA     TEMP             /SAVE IT
         JEQ      OUT              /IF ARG=0 DISABLE ADC
         FSUB     FC1              /LESS THAN 1 HZ?
         JLT      SMALL            /YES, FIND NEG. EXP
         FSUB     FC9              /NO, GREATER THAN 1HZ?
         JGE      LARGE            /YES, FIND POS. EXP
         FADD     FC10             /RESTORE VALUE
MDET,    FSUB     FC2              /LESS THAN 2?
         JGE      TWO              /NO, CONTINUE CHECKING
         FLDA     FC1              /YES, SET M=2
         JA       OUT
TWO,     FSUB     FC3              /LESS THAN FIVE?
         JGE      FIVE             /NO, GO SET TO 5
         FLDA     FC2              /YES, SET M=2
         JA       OUT
FIVE,    FLDA     FC3              /BECOMES M=5 IN 8 CODE
OUT,     ALN      0
         FSTA     MB               /PASS TO 8 CODE
         FLDA     EXP
         ALN      0
         FSTA     EXP8             /PASS TO 8 CODE
         FCLA
         TRAP4    #FREQ
         FCLA
         JA       FRQRTN
LARGE,   FLDA     FC1              /* TO DETERMINE EXP>0
         FADDM    EXP
         FLDA     TEMP
         FDIV     FC10
         FSUB     FC10
         JLT      LDET
         FADD     FC10
         FSTA     TEMP
         JA       LARGE
LDET,    FADD     FC10             /RESET, AND GO DETERMINE M
         JA       MDET
SMALL,   FLDA     EXP              /* TO DETERMINE EXP<0
         FSUB     FC1
         FSTA     EXP
         FLDA     TEMP
```

```
        FMUL    FC10
        FSUB    FC1
        JGT     SDET
        FADD    FC1
        FSTA    TEMP
        JA      SMALL
SDET,   FADD    FC1             /RESET, AND GO DETERMINE M
        JA      MDET
FC1,    F 1.
FC2,    F 2.
FC3,    F 3.
FC9,    F 9.
FC10,   F 10.
/       PROGRAMMER: E. H. PALS
/
/       PROGRAM FUNCTION: SET UP ROUTINE FOR FAST
/       DATA ACQUSITION.
/
/       DATE: 11/30/76
/       FILENAME: EPSET.RA
/       VERSION: *1A
/
/       METHOD: THIS PROGRAM IS A MODIFIED VERSION OF THE
/       RALF LISTING FILE OUTPUT OF FORTRAN IV COMPILER.
/       THE ORGINAL FORTRAN FILE IS SAVED UNDER THE HEADING
/       OF EPSET.F4.
/
/       NOTE: THIS ROUTINE NEEDS EPADC8.RA WHICH IS AN 8MODE
/       MODULE TO EXECUTE PROPERLY. IT IS IN THIS ROUTINE
/       THAT THE EXTERNALS REFERENCED IN THIS PROGRAM
/       ARE DEFINED.
        EXTERN  AR      /TELL LOADER VARIABLE IS NOT IN THIS MODULE
        EXTERN  MMNUM8  /SAME FOR THIS VARIABLE
        EXTERN  FLOAT
        SECT    SET
        JA      *ST
*XR,    ORG     .+10
        TEXT    +SET  +
*RET,   SETX    *XR
        SETB    *BASE
        JA      .+3
*BASE,  ORG     .+6
ARRAY,  ORG     .+3
NUM,    ORG     .+3
IERR,   ORG     .+3
        ORG     *BASE+30
        FNOP
        JA      *RET
        FNOP
*GOBAK, 0;0
*ARGS,  ORG     .+3
XNUM,   ORG     .+0003
*TMP,   ORG     .+0011
*LIT,   0000
        0000
        0000
        0001
        2000
        0000
        0002
        2000
        0000
        0002
        3000
        0000
```

```
          0003
          3000
          0000
          0013
          3720
          0000
          #LBL=.
          ORG       #LBL
#RTN,     BASE      #BASE
          JA        #GOBAK
#ST,      STARTD
          0210
          FSTA      #GOBAK,0
          0200
          SETX      #XR
          SETB      #BASE
          LDX       0,1
          FSTA      #BASE
          FSTA      #ARGS
          FLDA%     #BASE,1+
          FSTA      AR              /PASS THIS ADDRESS TO 8CODE
          FSTA      ARRAY
          FLDA%     #BASE,1+
          FSTA      NUM
          FLDA%     #BASE,1+
          FSTA      IERR
          STARTF
          LDX       0002,0
          LDX       0003,0
          FLDA      #LIT+0000
          FSTA%     IERR
          LDX       0004,0
          FLDA%     NUM
          FSUB      #LIT+0017
          JLE       #C0001
          FLDA      #LIT+0003
          FSTA%     IERR
#C0001,   LDX       0005,0
          FLDA%     NUM
          FSUB      #LIT+0000
          JGT       #C0002
          FLDA      #LIT+0006
          FSTA%     IERR
#C0002,   LDX       0006,0
          FLDA%     IERR
          FSUB      #LIT+0000
          JEQ       #C0003
          EXTERN    #NE
          JA        #RTN
#C0003,   LDX       0007,0
          FLDA      NUM
          STARTD
          FSTA      #C0004
          STARTF
          JSR       FLOAT
          JA        .+0004
#C0004,   JA        .
          FSTA      XNUM
          FNEG                      /MAKE IT NEGATIVE
          ALN       0               /CHANGE TO 8MODE COMPATABILITY
          FSTA      MMNUMB          /PASS VALUE TO 8CODE
          LDX       0010,0
          EXTERN    #NE
          JA        #RTN
          LDX       0012,0
```

```
        EXTERN  #NE
        JA      #RTN
/
/       THIS SECTION INITIATES THE DATA ACQUSITION PROCESS
/
        EXTERN  ADC8
        SECT    CO
        JA      #STCO
        0
        BASE    0
#STCO,  STARTD
        FLDA    30
        FSTA    #CORN,0
        FCLA
        STARTF
        TRAP4   ADC8
        FCLA
        JA      #CORN
```

```
LOADER V23  09 /14 /77
SYMBOL VALUE LVL OVLY
A      13417  0  00
ABS    17372  0  00
ADCRTC 11267  0  00
ADC8   10600  0  00
ADPLOT 32000  2  00
AINT   17242  0  00
ALOG   17515  0  00
ALOG10 16133  0  00
ANYCHR 16714  0  00
AR     10670  0  00
ARGERR 00204  0  00
AXIS   25445  1  03
AXISS  32745  3  11
B      13615  0  00
BACKB  11713  0  00
BELL   10101  0  00
BURNDM 32400  3  03
CBLK   14756  0  00
CGET   20165  0  00
CHAR   12600  0  00
CHARS  20066  0  00
CLOCK  15654  0  00
CLTIME 16066  0  00
CPUT   20175  0  00
DAC    10200  0  00
DARK   32400  3  02
DATA   20400  1  02
DATE   17066  0  00
EXIT   00223  0  00
EXPB   10720  0  00
FLAG   11000  0  00
FLOAT  15353  0  00
FLOW   20400  1  01
FREQS  15143  0  00
FUNC   24366  1  04
GO     15634  0  00
GRAPH  27175  1  03
GRAPHS 35400  5  01
HEAD   32400  3  10
IABS   17372  0  00
IDEB   15011  0  00
IFIX   17242  0  00
INPUT  20400  1  00
INT    17242  0  00
IPLOT  17273  0  00
ISIGN  20027  0  00
ISIM   16460  0  00
LABEL  26336  1  03
LABELS 32400  3  12
LIMIT  32400  3  00
LOOKUP 17027  0  00
MMNUMB 10677  0  00
MB     10723  0  00
ONQB   11510  0  00
ONQI   11400  0  00
OUT    12400  0  00
PARIN  32400  3  06
PLINT  23117  1  03
PLINTS 32400  3  11
PLT    23464  1  03
PLTS   32400  3  13
POS    10400  0  00
POSDEM 33104  3  03
```

```
POSITN 27402  1  03
POSITS 35400  5  00
POT    20400  1  03
PT     24501  1  03
PTS    34400  4  00
RESPND 32400  3  05
ROTATE 16765  0  00
RUN    32400  3  07
SAMPD  16370  0  00
SAP    32714  3  00
SAPDEM 32400  3  04
SAPINT 11600  0  00
SET    15372  0  00
SFL2   32400  3  01
SIGN   17754  0  00
SMXNM  22136  1  04
SMXOPT 20400  1  04
SMXSUB 20400  3  05  EX*
SPBLK  16672  0  00
SUBONE 13000  1  02  EX*
SUBTHR 13000  0  00  EX*
SUBTWO 13000  1  02  EX*
SWITCH 10000  0  00
TIME   16105  0  00
TUNE   33121  3  04
TUNE8  11671  0  00
 MAIN  13000  0  00
 36000  = 1ST FREE LOCATION
LVL OVLY LENGTH
 0  00    20352
 1  00    10416
 1  01    03342
 1  02    07660
 1  03    07351
 1  04    11116
 2  00    00370
 3  00    01260
 3  01    00275
 3  02    01162
 3  03    01224
 3  04    00737
 3  05    01102
 3  06    01616
 3  07    01163
 3  10    01651
 3  11    01236
 3  12    00637
 3  13    01015
 4  00    00744
 5  00    00347
 5  01    00205
```

APPENDIX B

Simple Hardware Control Approach for Sequencing Chemical Instruments

S. R. Crouch,[1] D. N. Baxter,[2] E. H. Pals, and E. R. Johnson[3]
Department of Chemistry
Michigan State University
East Lansing, Mich. 48824

Simple Hardware Control Approach for Sequencing Chemical Instruments

In modern analytical instrumentation, digital controllers have become more and more complex and flexible as instruments have become more highly automated. The controller directs the sequencing of various instrument functions. For example, in an automated gas chromatography system, the controller might be required to send signals to an automatic sampler for introduction of the sample into the column, to control the oven temperature and its temperature program, to control the sensitivity of the detection system, and to control the detection and integration of the various chromatographic bands as they elute from the column. Such controllers may be simple fixed order sequencers or may require branching to skip certain functions if certain instrument conditions are set by the user (i.e., unless the temperature program button and rate have been set, the GC controller would skip this part of its sequence).

In modern instruments, controllers take the form of hardwired logic systems, minicomputers or microprocessors. The microprocessor is certain to become the controller of choice in the future for many operations, because the program (sequence) can be altered without changes in hardware. Also, even the most complex control operations, with multiple and nested branches, can be readily carried out under the supervision of a microprocessor. However, the microprocessor requires the instrument designer to be re-educated in terms of software and processor capabilities. This education process may require months to years in order to get working systems into the laboratory with "intelligent" instrument controllers.

A few years ago our research group became heavily involved in controllers when we decided to undertake a project that involved the spatial profiling of atomic concentration above a filament-type electrothermal atomizer using atomic absorption spectrometry. We desired to automate fully the operations of sample introduction to the filament, movement of the filament in both x and y directions away from the AA optical axis to predetermined destinations, temperature programming of the filament, data acquisition of background-corrected peak area, and automatic return to position "zero" for pickup of a new sample. The filament position controller was intended to move the atomization cell freely over a total displacement of 1 in. in either of two dimensions, through the use of stepper motor-driven micrometer translation stages. The controller was required to function in either a "local" mode, in which the desired filament position could be set by the user via front panel BCD switches, or in an "on line" mode in which the controller received binary information from a minicomputer and functioned as a hardware slave to the processor. In addition, we required the controller to convert BCD position code to binary code in the local mode, to use position instead of displacement as the input (i.e., to remember where it is and to determine the direction to move and how far to go), and to be capable of a hardware reset to an optically defined reference position. Finally, we required the controller to correct for backlash in the leadscrews of the translation stages in that if a requested movement was in the positive direction (away from the optical reference), the controller would intentionally move the cell to a known distance beyond the desired destination and then return to the true destination, in contrast to movement in the negative direction, which would proceed directly to the destination without such overshoot. In this manner, no matter whether the net movement was positive or negative, the cell would be moved last in the negative direction, thus leaving the leadscrews tightly meshed in that direction of movement.

Because of the complexity of our application, we did not look forward to relearning combinatorial and sequential logic methods, component minimization techniques, and other forgotten digital design methods. Microprocessor approaches were considered but later abandoned because the interleaving of real-time tasks each with their own timing requirements put the problem on a level of complexity where our then meager microprocessor experience could not hope to reach in a reasonable time. Fortunately, the sinking feeling we had originally experienced was short-lived when we discovered the simple approach to the design of complicated controllers described in this paper. Even for our complex application, this systems-level approach to controller design enabled us to design in about 3 h a unit requiring over 50 IC's per motor controller. Construction and debugging of the position controller required additional time, of course. The speed of our design was made possible by the use of a method, originally described by Richards (*1*), that allows all the complex sequential operations to be treated by using only a simple flow chart description at the beginning and by implementing the flow chart with 3-4 MSI IC's.

To understand the approach, we present in the first section of this paper the general concepts and principles of the "state" of a controller (*2*) and the simple implementation that results from the Richards (*1*) approach. We then present a chemical example of a "fall-through" sequencer

---

[1] To whom correspondence should be addressed.
[2] Present address, Eastman Kodak Co., Kodak Park, Building 82, Rochester, N.Y. 14650.
[3] Present address, Department of Chemistry, Wayne State University, Detroit, Mich. 48202.

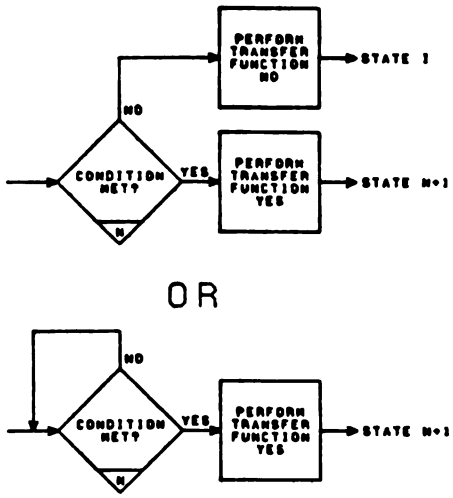


Figure 1. Basic flow chart units for transfer conditions and transfer functions. N: state number

that controls an automatic fraction collector used in elution chromatography. In the last section we describe a branching sequencer and illustrate its use as an electrothermal atomizer position controller.

Concept of the State

Automatic sequencing of measurement and control operations is highly desirable for modern chemical instruments. We assume here that the operations to be sequenced are ON-OFF operations that can be controlled by logic level signals. The sequencer must then produce logic level outputs for each controlled operation and must be responsible for the timing relationships of the appropriate logic level signals. There are a variety of approaches to the design of sequencers (*3*). However, the systematic approach presented here allows even the most complex sequencing applications to be handled with ease.

To help understand this approach, several terms must be defined. This can be done most conveniently with the aid of an example. Consider the controller for the thermostat on a temperature bath. The method described by Richards (*1*) considers it to have three states: State 0 = thermostat off–heater off; State 1 = heater on; and State 2 = heater off. In State 0 the thermostat controller waits for the condition of the thermostat power switch to be on. The on condition of the switch then is referred to as a *transfer condition*. When the transfer condition for State 0 is true, i.e., the switch is on, the controller advances to State 1. The process of this advance causes a *transfer function* to occur. The heater is turned on. The second state (State 1) tests to see if the bath temperature exceeds the limit set on the thermostat. When the bath temperature does exceed the limit, the controller steps to State 2, which causes another transfer function to occur. The heater is turned off. The transfer condition for State 2 is a bath temperature below the thermostat limit. When this condition is met, no transfer function is required except to return to State 0 and repeat the cycle.

Viewing such a trivial example in terms of states, transfer conditions, and transfer functions may seem overly complicated, but it is useful for two reasons. First, the method is also applicable to much more complex control applications. Second, with this approach the circuit design is greatly simplified by use of a standard circuit for the heart of the controller.

The application of this approach involves first converting the states, transfer conditions, and transfer functions into a standard flow chart format. The two basic flow chart units for each state are shown in Figure 1. The unit used depends on what is to occur at that particular state. Either unit contains a diamond-shaped box that represents the state and contains the transfer condition. Depending on the application, the basic unit will also contain one or two rectangles that represent the transfer functions which are to occur when the transfer conditions are met or not met. In this method the transfer function on the "no" leg (if one exists) is available only if the transfer condition not being met causes a move to a State I other than the present one. This transfer function, if it is used, is called a *secondary transfer function*, as opposed to the *primary transfer function* on the "yes" leg. Figure 2 shows the flow chart for the bath thermostat example. No secondary transfer functions are required for this application; therefore, the controller is an example

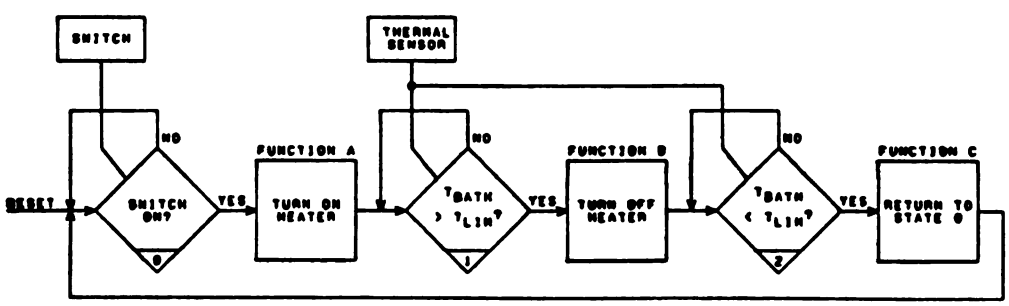


Figure 2. Sequencer flow chart for bath thermostat example

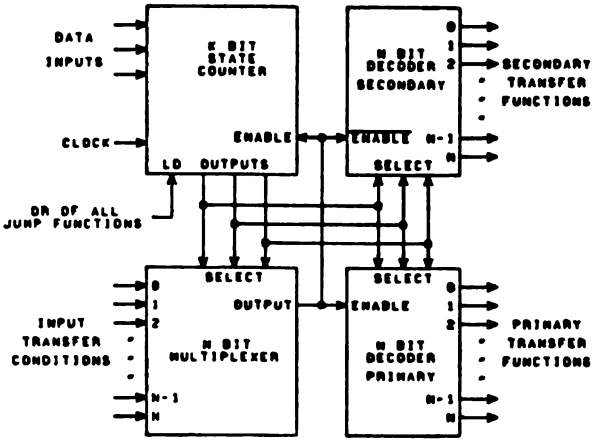


Figure 3. General schematic diagram of controller

of a *fall-through sequencer* (no conditional branches).

The "heart" of a sequencer based on this approach can be implemented with the standard integrated circuits shown in Figure 3. From the flow chart, the number of states in the sequence and the various transfer conditions and functions are known. If no secondary transfer functions are necessary, the secondary n-bit decoder can be eliminated. Thus, the heart of a sequencer consists of a k-bit counter, an n-bit multiplexer, and one or two n-bit decoders, where n is the number of states in the sequence and $n \leq 2^k$.

The k-bit counter keeps track of the state of the controller. Logic level signals representing the various transfer conditions are connected to the n-bit multiplexer. The outputs of the state counter determine which transfer condition is supplied by the multiplexer to one or two n-bit decoders. The multiplexer output also allows the state counter to increment to the next state on the next clock pulse after a "true" transfer condition. The presence of a "true" transfer condition at the multiplexer output causes the appropriate primary transfer function (as selected by the state counter) at the output of the decoder to go true. However, only a pulse occurs on the selected decoder output line, because the next clock pulse increments the state counter to the next state. If the transfer condition is false, either a secondary transfer function can be generated (at the secondary decoder output), or the sequencer merely waits in the same state until the transfer condition becomes true. To provide the capability for the sequencer to skip certain states under the appropriate conditions, a state counter capable of parallel loading is used. When a jump or branch in the sequence is necessary, the load (LD) line of the state counter is activated, and the new state is loaded into the counter through the data inputs.

An eight-state sequencer can be readily implemented with a 74163 counter, a 74151 multiplexer, and one or two 7442 decoders. At current single unit prices, they can be obtained for just over $3.00.

Fall-Through Sequencer

The fall-through sequencer is the simplest type. In this sequencer the only decision to be made is whether to remain in the present state or proceed to the next state. In a closed-loop operation this decision is based on receiving an "all done" signal (flag) from a previously actuated circuit. In an open-loop operation the change of state may be immediate or may occur upon receiving a signal that a time delay has elapsed.

A chemical example of a fall-through sequencer is a controller for an automatic fraction collector used in elution chromatography. Typically, the fraction collector has a rotatable tray that contains vials sequentially rotated beneath the column outlet valve. When a vial is directly beneath the outlet valve, the valve is opened for a preset time, then closed, and the next vial rotated into the receiving position. These motions are repeated until the desired number of fractions has been collected.

In Figure 4 the decisions and actions involved in controlling this apparatus are represented in flow diagram form, along with the external inputs used to make the decisions. Assume that the operator has initialized the apparatus, by rotating the first vial into position beneath the valve, and loaded the number of fractions desired into a counting register. Also, assume that the controller is initially in State 0. Condition A is satisfied since the counter is nonzero; therefore, the controller generates Function A, which is to open the delivery valve and to start the delay timer. Now the controller moves to State 1 and repeatedly queries the delay timer until it signals that enough solution has been de-

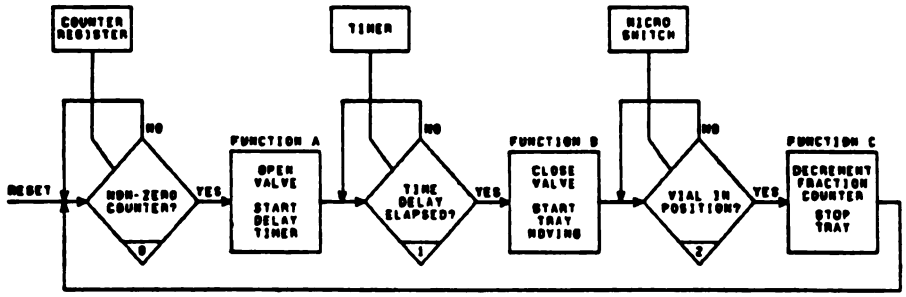


Figure 4. Sequencer flow chart for automatic fraction collector

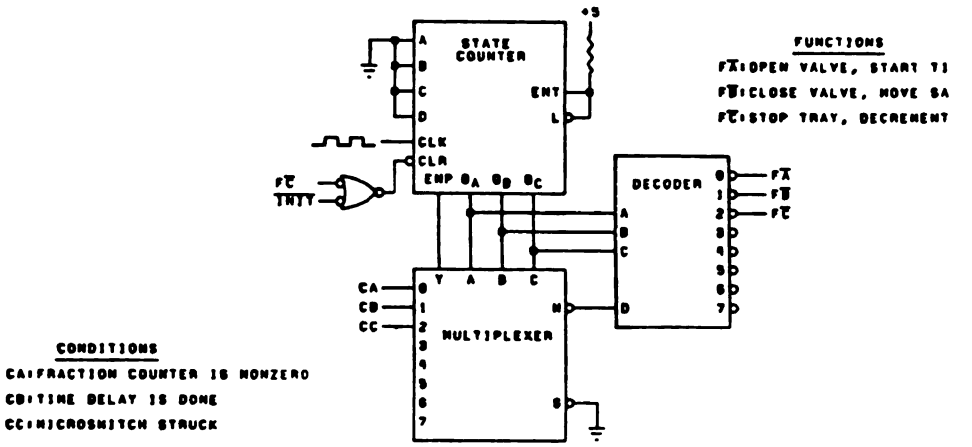


Figure 5. Schematic diagram of fraction collector controller

livered (Condition B). When Condition B has been satisfied, the controller closes the valve and starts the vial tray moving (Function B). Now the controller moves to State 2 and waits for a position indicator microswitch beneath the outlet valve to be tripped (Condition C). When Condition C is met, the controller must decrement the fraction counter register, stop the tray, and effect a jump back to controller State 0 (Function C). This sequence is then repeated until the fraction counter register is decremented to zero. When this happens, Condition A can never be met. The apparatus halts since it becomes hung up in State 0.

This controller can be quite easily implemented with three IC's: a 74163 counter, a 74151 multiplexer, and a 7442 decoder. Of course, additional packages may be necessary for such things as delay timing, but these three IC's handle the generation of all the signals necessary to initiate the actions at the appropriate times. Figure 5 shows how these IC's are interconnected and also the input and output signals required in this particular application. When power is turned on, an initialize pulse ($\overline{\text{INIT}}$) is required to clear the fraction counter register and to assert the CLEAR (CLR) input of the state counter. This forces the state counter to State 0. The Q_A, Q_B, and Q_C outputs of the state counter are all LO (zero), which sets the multiplexer's Y output LO. This LO signal is fed to the state counter's ENABLE P (EN P) input, which inhibits the counter from counting clock pulses at its CLOCK (CLK) input. Thus, it remains in State 0. The multiplexer's W output (the complement of the Y output) is directed to the decoder's D input. Output W is HI, and examination of the truth table for the decoder indicates that the 0–7 decoder outputs must be HI when D is HI (LO true logic is used at these outputs so that a function is generated only when the output goes LO). This means that initially none of the functions is being asserted. When the operator gives the "GO" command, the number of fractions desired is loaded into the fraction counter register, and Condition A is satisfied. Now the multiplexer's W output goes LO, and the decoder points to a decimal 0, which asserts the Function A signal ($\overline{\text{FA}}$).

Function A triggers external circuitry to open the delivery valve and start a time delay circuit. In addition, the multiplexer Y output is HI, so that the state counter's ENABLE P input is asserted, and on the first clock pulse the counter is incremented once (State 1). Now the multiplexer shifts from selecting data channel 0 (Condition A) to data channel 1 (Condition B). If the required time delay has not yet elapsed, Condition B is not yet satisfied so that the multiplexer Y output falls, locking the state counter in State 1, and the W output goes HI, concluding function A and inhibiting the decoder from generating any functions.

When the required time delay has elapsed, Condition B is satisfied, and a similar sequence occurs in the state counter, multiplexer, and decoder. The multiplexer's W output enables the decoder, and its A, B, C, and D inputs point to decimal 1. Thus, output 1 (Function B) is asserted. This triggers the valve to close and starts the tray moving. The multiplexer Y output goes HI, allowing the counter to count one pulse and move to State 2. Now the state counter requests the multiplexer to address the status of Condition C.

When the new sample vial reaches the position beneath the outlet valve, a microswitch is tripped and Condition C is satisfied. Again this activates the decoder, and Function C is asserted. Function C stops the tray and decrements the fraction counter register. Also, since this is the last function in the sequence, it is used to assert the CLEAR input of the state counter and return the controller to State 0.

Unless this was the last fraction to be collected, the fraction counter register will be nonzero when the state counter enters State 0 so that Condition A is immediately satisfied. Thus, as soon as the Function C pulse has cleared the state counter to State 0, Function A will be generated, and on the next clock pulse State 1 will be achieved. This entire delivery sequence will continue until the fraction counter register reaches zero, which prevents Condition A from being satisfied. Then the apparatus will halt because the controller becomes hung up in State 0 until the operator intervenes to start a new collection sequence.

Branching Sequencer

In many cases, it is desirable to have a sequencer capable of branching; that is, one in which sections of the basic state sequence may be skipped, repeated, or otherwise altered as requested by the instrumentation under control. An example of such a sequencer is the electrothermal atomizer position controller mentioned previously.

The flow diagram of its sequence is shown in Figure 6. The system waits in State 0 until data representing a new location are received, and a command to move is given. Upon receipt of this command, the overflow flag is cleared, and the system determines if it is being operated under local or computer control. If control is local, the desired destination has been entered in BCD format from front panel thumb-wheel switches, which requires a BCD to binary conversion. A check is made to see if the system is free of overflow; if so, another check is made to see if BCD to binary conversion is complete. If conversion is not complete, the registers are clocked, and the checks for overflow and complete conversion are repeated. If an overflow should occur, the sequence will abort to State 0 with the overflow flag set. (Overflow results from entering a destination outside the defined limits of the system.) If no overflow occurs, then upon completion of the conversion, the stepper motor direction is determined, and the stage begins to move. If control is from the computer, the BCD to binary conversion is unnecessary. Thus, this section of the sequence is omitted.

The stage moves until the destination is reached. At this point the system checks to see if the direction of movement was positive. If so, the stage continues to move 10 additional increments positive, and when finished, reverses direction and comes back 10 increments negative. If movement was originally negative, the sequence is concluded immediately. In this manner, gear play is removed from the mechanism by assuring that movement always occurs last in the negative direction.

To implement a branching sequencer such as this, it is necessary to add only one additional decoder IC to the basic controller circuit. The actual wiring is shown in Figure 7. The actions of this controller are identical to the simpler three-chip system previously discussed as long as the basic, nonbranching sequence is followed. If a branch is called for, however (if the selected data channel of the multiplexer is LO when a potentially branchable state is reached), the following events occur. The LO state of output Y from the multiplexer causes the function pulse to be generated by the second decoder instead of the first decoder. This function pulse is used by the instrument to perform tasks in the same manner as any function pulse from the first decoder. In addition, this pulse is also fed back to the state counter where it serves simultaneously to provide binary data representing the state to which to branch as well as to load those data into the state counter. For example, in the positioner sequence, if the system is under computer control when State 1 is entered, the LO level at the data Channel 1 of the multiplexer disables the first decoder, but causes the second decoder to generate Function B2. Function B2 is used in the circuitry to update the direction of movement and start the motor, just as Function D1 of the first decoder does under local control. But it is also present at the C and Load (L) inputs of the state counter so that as it occurs, the state counter is forced to skip directly to State 4 through parallel loading. All additional functions that are part of other such branches are similarly directed back to the Load and appropriate data inputs of the state counter, by use of simple gates as shown to OR them together as required.

By adding additional multiplexer and decoder chips to the controller, it is possible to produce sequencers of even greater complexity in which the branches consist not only of single

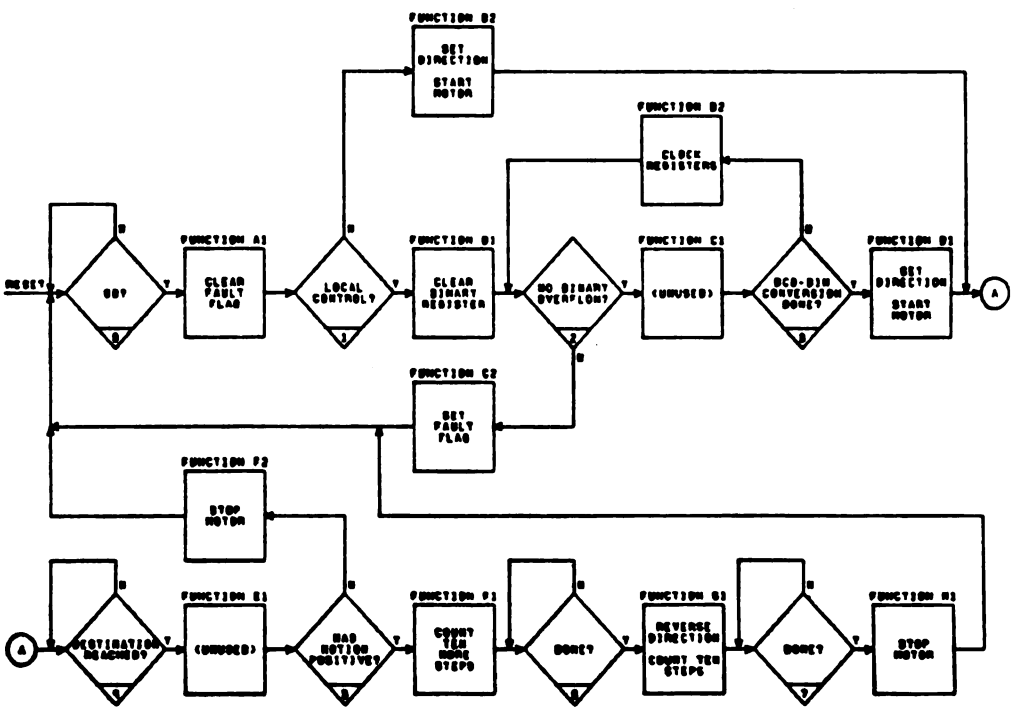


Figure 6. Sequencer flow chart for filament positioner

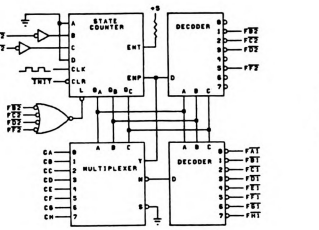


Figure 7. Schematic diagram of filament positioning controller

function pulses, but also of additional states or even complete subsequences reminiscent of nested computer subroutines (*1*). Similarly, it is also possible to substitute basic chips different from those employed here to produce sequences of more than eight basic states. However, such systems can quickly become unwieldy in their complexity and might very well be better performed using microprocessor-managed controllers even if the user has no previous experience with such devices (*4–6*).

Conclusions

We have described in this paper a hardware approach for designing instrument sequencers. We feel that this approach greatly reduces design time, yet allows complex sequences to be readily implemented. The increasing education of chemists as to the potential, the software, and the interfacing of microprocessors will certainly lead to a level of controller complexity at which a microprocessor is the sequencer of choice. Even so, the concepts presented here should enable simple controllers to be designed and implemented readily by chemists unfamiliar with microprocessors or for applications in which it is undesirable to "tie up" the processor in simple, unchanging control tasks.

In addition, the hardware approach is advantageous for high-speed control functions and for applications that involve the interleaving of tasks with quite dissimilar timing requirements. In fact, we have found in our own laboratories that hardware sequencers are important adjuncts to instrumentation systems supervised by minicomputers or microcomputers. Besides the electrothermal atomizer positioning systems, we have used this approach to design a minicomputer data acquisition system and a communications network between several microprocessors and a minicomputer. The simplicity and power of the approach should prove valuable to other chemists involved in the automation of chemical instrumentation.

Acknowledgment

We are grateful to T. V. Atkinson for his assistance in the use of a computerized graphics facility for the production of our figures.

References

(1) C. L. Richards, *Electronics*, **46** (3), 107 (1973).
(2) T. R. Blakeslee, "Digital Design with Standard MSI and LSI", pp 117–20, Wiley-Interscience, New York, N.Y., 1975.
(3) H. V. Malmstadt, C. G. Enke, and S. R. Crouch, "Electronic Measurements for Scientists", pp 662–7, Benjamin, New York, N.Y., 1973.
(4) D. R. McGlynn, "Microprocessors", Wiley-Interscience, New York, N.Y., 1976.
(5) B. Soucek, "Microprocessors and Microcomputers", Wiley-Interscience, New York, N.Y., 1976.
(6) T. R. Blakeslee, "Digital Design with Standard MSI and LSI", pp 150–232, Wiley-Interscience, New York, N.Y., 1975.

D.N.B. was recipient of both a summer and full-year ACS Analytical Fellowship under the sponsorship of Procter & Gamble Co., Cincinnati, Ohio.

S. R. Crouch

D. N. Baxter

E. H. Pals

E. R. Johnson