ON AND OFF-LINE COMPUTER AUTOMATION FOR CHEMICAL ANALYSIS

USING A NUCLEAR DATA 812 MINICOMPUTER

by

John P. Hackney

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in the

Chemistry

Program

John D Van Norman 6/1/77

Advisor Date

Dean of the Graduate School Date

YOUNGSTOWN STATE UNIVERSITY

June, 1977

ABSTRACT

ON AND OFF-LINE COMPUTER AUTOMATION FOR CHEMICAL ANALYSIS USING A NUCLEAR DATA 812 MINICOMPUTER

John P. Hackney

Master of Science

Youngstown State University

This paper is meant to serve two basic purposes: 1. It is a complete report of the research done by this writer on this topic, and 2. it is meant to serve as a primer, of sorts, for the chemist with little, or no, background in computer technology. In this respect, it is hoped that the language and detail of this paper will enable any chemist to pick up this project and expand the research without having to spend a considerable amount of hours deciphering seemingly indecipherable instruction manuals.

The introduction of this paper serves to point out the advantages and disadvantages of clinical laboratory automation. These topics are discussed in some detail.

Following the problem definition, a section of hardware descriptions is also included to acquaint the unfamiliar reader with the apparatus used during this research.

A detailed discussion of computer languages, including Machine Language, Assembly Language and Nutran, is given with corresponding examples of programming in each language.

Finally, two alternate solutions of the problem are offered; one for off-line and the other for on-line operation.

A complete listing of all programs used in this research, as well as loading and initialization instructions for these programs and a glossary of terms are included in the appendices.

ACKNOWLEDGEMENTS

I sincerely thank all those who have helped me during the course of this research, especially, the Mohawk Area School Board and Nuclear Data, Inc. for their selfless assistance and cooperation.

My sincerest and deepest thanks and appreciation go to my advisor, Dr. John Van Norman, under whose guidance I am proud to have studied. For the wealth of information I acquired through him, his time, patience and understanding, I am eternally grateful and indebted to him.

Finally, but most importantly, I would like to thank my parents. Without their encouragement, sacrifices, belief in the value of education and, especially, love, I never would have made it. To them, I owe a debt that can never be repaid.

TABLE OF CONTENTS

LIST OF SYMBOLS

SYMBOL	DEFINITION	UNITS OR REFERENCE
a	Constant	none
a_0	Intercept	See (Eq. 5)
a_1	Slope	See (Eq. 4)
A	Absorbance	See (Eq. 1)
b	Path Length	cm
c	Concentration	mg/100 ml
K	1,024 Computer Words	none
n	Number of Items	none
r^2	Coefficient of Determination (Reliability of Data)	See (Eq. 6)
%T	% Transmittance	none
T	Transmittance	none
x	Glucose Concentration	See (Eq. 4,6)
$\overline{x}$	Average of all Glucose Concentrations	See (Eq. 5)
y	Absorbance	See (Eq. 4,6)
$\overline{y}$	Average of all Absorbances	See (Eq. 5)
Σ	Summation of the mathematical terms that follow	none
$>$	Quantity preceding symbol is greater than quantity following symbol	none
σ	Standard Deviation	none

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

A modern clinical laboratory is called upon daily to perform numerous analyses, computations, and reports, all in the name of medical care for the patients. As medical science has advanced in recent years, and as analytical techniques have been improved, so too, the number of test requests received by the clinical laboratory has increased proportionately.

> The workload of the clinical laboratory is significantly affected by the rapid rate of change in the practice of medicine. Studies at civilian systems indicate that without increasing the number of patients, the clinical laboratories are evidencing a compound annual rate increase in the number of tests requested of approximately 15 percent. This represents a 100 percent increase in the workload approximately every six years.[1]

Clinical laboratories, in the last fifteen years, have become the heart of most hospital systems, and, as it is with the heart of most organisms, when it is over worked, the entire system may break down. It is a well-known fact that an obese person will be advised by his physician to shed excess weight, thus relieving much of the burden on his heart; so too, when a clinical laboratory has become overworked, the workload should be alleviated, not by the reduction of the number of procedures performed, but by the automation of existing methods.

Justification of Automation

The introduction of automation to a clinical laboratory is not without justification. It has been shown that the automation of a clinical laboratory will show: reduction in cost; increased lab billings; improved patient care; and reliability.

Reduction in Cost

By reducing or eliminating the clerical duties that the analyst must perform, a significant amount of time can be saved, thus freeing the analyst for more tests and also diminishing the amount of clerical errors made while recording data and/or results.

> For example, the 1080 system relieved the Technicon analyst of reading, recording, and calculating data from strip chart recorders. This reduced his working time about 20 min./hr. Working time on other jobs was similarly reduced by computer calculations, as compared to manual work on desk calculators. Consequently, jobs could be combined and handled by fewer people.[2]

Reducing the number of analysts obviously would bring monetary savings to a clinical laboratory. However, by maintaining the same work force, but utilizing automated techniques, each technician is able to increase his/her output significantly, generally without an increase of physical effort, and, in many cases, the job becomes less laborious.

> Further analysis indicates that approximately 80 percent of the clinical laboratory's operating budget relates to personnel costs, and that current design criteria generally allows for automation of only 35 percent of all tests....Introducing new technology will not only show a cost benefit through reduced total labor or increased throughput, but will also provide the capability for more readily absorbing the enormous problem of increased workload.[3]

The analyst is able to perform a substantial amount of additional tests per month. In a typical clinical laboratory, costs per test generally decrease as the volume increases. Table 1 shows the comparative studies made at Beaufort Naval Hospital and Ft. Dix Army Hospital.[4]

Particularly, refer to the cost analysis for the single channel AutoAnalyzer. While Ft. Dix Hospital performs approximately 4.4 times the number of manual tests per month as compared to Beaufort Naval Hospital (3027:684, Column 2), there is a similar significant trend in the Cost/Month for manual tests at each hospital. Ft. Dix is listed as having 4.2 times the Cost/Month as does Beaufort (1526: 381, Column 3). When both hospitals utilize automated systems, while the number of tests/month is still the same for both hospitals, there is a difference in Cost/Month between the two hospitals of only one dollar (234:233, Column 4). This represents savings of $1,292 at Ft. Dix Hospital, and $148.49 at Beaufort Hospital.

In other words, by using a single channel AutoAnalyzer, a clinical laboratory can quadruple its number of tests per month with only a negligible increase in the monthly cost. At the same time, the additional monies saved by the elimination of these particluar manual tests are an obvious benefit of automation.

It has been shown above that reduction in cost is largely due to reduction in time per test. Perhaps a closer look at the various steps in an analysis, as shown in Table 2[5], will better illustrate just how automation can save valuable time while reducing and/or eliminating many inconsistent errors.

One will notice, from studying Table 2, that the sample, once reaching the laboratory, is untouched by human hands. The technician

TABLE 1

POTENTIAL SAVINGS (+) PER MONTH FROM AUTOMATION AT TWO MILITARY CLINICAL LABORATORIES[4]

BLOOD CHEMISTRY-BEAUFORT

	Total No. Tests/Mo. Automatable	Manual Cost/Mo. Of Automatable Tests	Automated Cost/Mo.	Automated Savings[b]
DSA-560	1606	$ 959	$1593	$ -634
Hycel	1432	838	1707	-869.42
SMA 12/60	1557	873	3557	-2684
Dupont ACA	1379	887	3340	-2453.60
Centrifi-Chem	1321	515	757	-242
AutoAnalyzer (Single)	684	381	233	+148.49
AutoAnalyzer (Dual)	684	381	719	-338
BLOOD CHEMISTRY-DIX				
DSA-560	8146	$4803	$2242	$+2561
Hycel	7104	4195	3037	+1158
SMA 12/60	8071	4516	3874	+642
Dupont ACA	7568	3619	8156	-4537
Centrifi-Chem	6670	3969	1143	+2826
AutoAnalyzer (Single)	3027	1526	234	+1292
AutoAnalyzer (Dual)	3027	1526	864	+662

[b]The tables above show potential savings (+) or additional expense (-) which would be anticipated through automation at Beaufort Naval Hospital and Ft. Dix Army Hospital clinical laboratories.

merely has to acquire the sample, start the analysis, then sit back and wait for the results which are simultaneously printed in legible form and stored in the computer's vast, efficient memory.

Increased Lab Billings

The use of automation in clinical labs has also resulted in improved efficiency in the area of patient billing. At the Youngstown Hospital Association, when "a test result is entered into the patient's random access file, a charge is likewise generated..."[6] This immediacy of charge generation is of value for several reasons.

TABLE 2

POTENTIAL BENEFITS OF AUTOMATION[5]

Step No.[a]	Sample and data flow	Automated data handling	Potential benefits
1.	Collect sample, record accessioning data, ship to laboratory		
2.	Accession sample data into computer	Use optical character reader and scan form	Makes manual typing unnecessary.
3.	Analyze standard	Use on-line computer-controlled instrument	Eliminates all data-recording operations
4.	*Calibrate instrument; check calibration*	Compute curve of best fit, gradient, and correlation coefficient	Eliminates manual or calculator computations of curve of best fit, etc. Correlation coefficient gives quality control of calibration
5.	Analyze sample	Use on-line computer-controlled instrument	Eliminates all data-recording operations
6.	*Compute result*	Compute result from instrument response	Makes manual or calculator computations unnecessary
7.	*Check result for:*		
	a. significant figures	Compute number of significant figures	Avoids a manual data inspection step
	b. Gross errors	Check for gross errors; e.g., NH_3 should be $<$ total nitrogen	Ensures self-consistent results
8.	*Report results*	Use telephone link from mini-computer to central computer; print results.	Eliminates much of the manual typing
9.	Record results in permanent records	Central computer files on magnetic tape	Makes results available for later reference and statistical reports

[a]Steps set in italics are automated in the system reported here. All other steps (except 1) can also be automated, but such changes would yield lower initial benefits.

Since a test result cannot be entered into the patient's file without simultaneously generating a charge, late and missing charges are eliminated. The absent laboratory report will be called rapidly to the attention of the pathologist by the attending physician. This follow-up of the test result serves automatically also to check on the fiscal activities of the laboratory.[7]

(This is more fully discussed by Rappoport and Gennaro.[8])

This system allows for more efficient patient billing in that every charge generated by the patient during his stay in the hospital is tallied and recorded by the computer. Thus, the patient receives one bill and need not worry about any charges that may have been misplaced or overlooked for which he would be billed at a later date.

Consider also that some of these mishandled charges may never be found. This results in reduced receipts for the clinical laboratory. With the computer-aided billing as mentioned above these monies need never be lost. "These increased billings are substantial for some laboratories (over $10,000 per month in "found revenue")".[9]

Improved Patient Care

Perhaps the most significant improvement that occurs from the introduction of automated techniques into the laboratory is the notably faster reporting of results of tests. This reduction in "turn around time"[c] allows the attending physician to evaluate his patient's progress much more rapidly and, if necessary, order modifications in his schedule of treatment much sooner and with more confidence than was ever before possible.

"A second benefit is a reduction in missed specimen collections."[10] The computer will print out complete lists of instructions for the labor-

---

[c]See Glossary. (Appendix E)

atory's drawing team. "The instructions include the identification number of the patient, his hospital number, ward location, room and bed numbers, the numbers and types of specimens to be collected, and specific container sizes."[11]

In addition to the instruction lists generated by the computer, specimen container labels for each specimen to be collected are also produced by the computer. These labels show all information necessary to the proper identification of the patient and his specimen. When the technologist compares the available samples against the loading list, any missing samples are immediately evident. The missing specimen can then be traced to locate it within or outside the laboratory or even to find the reason why it is not there.

> A fourth patient care benefit derived from a laboratory computer system is the correct, convenient, and rapid interpretation of laboratory findings by the attending physician....The formats of computer generated reports, in general, are now easily comprehensible to the attending physician who had been acclimated to the old, familiar, manually-posted chart.[12]

These reports are generally produced in tabular form showing the results of each patient's tests, usually by wards, daily and, as at the Youngstown Hospital Association, each patient has his own weekly report which is updated daily, thereby showing an entire week's statistics at a glance (See Tables 3 and 4.).

Laboratory reports are of no value, however, if they are never received by the attending physician. Formerly, a lost report meant the repetition of a test, often at the expense of the patient's comfort and usually lengthening his stay in the hospital.

> The reduction in lost or missing patient reports is a patient benefit available from a laboratory computer system in that the computer records are in readily retrievable form. All the test

TABLE 3

WARD REPORT[13]

THE YOUNGSTOWN HOSPITAL ASSOCIATION

07/25/68 01.12 P.M

WARD 3 INTENSIVE CARE WARD REPORT

0327A ROWBOTHAM JAMES H		471957
RBC/WBC	M/TH C.MM	3.20/10.4
HGB	G%	10.1
PCV	%	30.0
CORP CONST-MCV		93.7
MCH		31.5
MCHC		33.6
DIFF-SEGS		73
STABS		01
EOS		01
LYMPHS		24
MONOS		01
PLATELETS		OK
GLUCOSE FAST	MGM%	143
0330A ADAMS JAMES		471000
ELCTRLYT CL	MEQ/L	100
ELCTRLYT CO2	MEQ/L	27
ELCTRLYT POTASS	MEQ/L	4.3
ELCTRLYT SODIUM	MEQ/L	134
ELCTRLYT PH SER		7.43
GROSS AND MICRO B		SEE REPORT
0332A SOUTHERLAND JAMES E		472051
RBC/WBC	M/TH C.MM	5.05/21.4
HGB	G%	13.8
PCV	%	40.5
CORP CONST-MCV		80.1
MCH		27.3
MCHC		34.0
DIFF-SEGS		84
STABS		06
LYMPHS		05
PLATELETS		OK
BUN	MGM%	7
ELCTRLYT CL	MEQ/L	101
ELCTRLYT CO2	MEQ/L	27
ELCTRLYT POTASS	MEQ/L	3.7
ELCTRLYT SODIUM	MEQ/L	135
ELCTRLYT PH SER		7.45

TABLE 4

PATIENT SUMMARY REPORT[14]

SOUTHERLAND JAMES E 11 HOPS. NO.472051-1 07/27/68 PAGE 1

TURNER J J MD

WARD 3 INTENSIVE CARE		SUNDAY 07/21/68	MONDAY 07/22/68	TUESDAY 07/23/68	WEDNESDAY 07/24/68	THURSDAY 07/25/68	FRIDAY 07/26/68	SATURDAY 07/27/68
HEMATOLOGY								
RBC/WBC	M/TH C.MM			4.85/8.7		5.05/21.4		4.35/25.6
HGB	G%			12.8		13.8		12.4
PCV	%			38.5		40.5		37.0
CORP CONST-MCV				79.3		80.1		85.0
MCH				26.3		27.3		28.5
MCHC				33.2		34.0		33.5
DIFF-SEGS				48		84		88
STABS				06		06		
EOS				01				
LYMPHS				45		05		07
MONOS				00		05		05
PLATELETS				OK		OK		OK
RBC SIZE				MICRO				
ABNOR MORPH				SL				
PTT	SEC			28.4				
BLOOD TYPE ABO				A				
BLOOD TYPE RH				POS				
TRANSFUSION-SEE REPT					3000ML			
URINALYSIS								
URINE-COLOR/APPEAR				STRW/CLER				
SPECIFIC GRAV				1.025				
PH				5.2				
ALBUMIN				NEG				
SUGAR				NEG				
OCCULT BLD				NEG				
EPITHELIAL				SQUAM 1+				
CHEMISTRY								
BUN	MGM%			11		7		

> values that have been done on the patient are available in the memory, so that if the report is physically lost or misplaced, values are not lost.[15]

Thus, another "pound of flesh" need not be taken from the patient in order to cover for a careless mistake on the part of the hospital staff. Though a relatively minor one, the recoverability of "lost" data is a decided advantage of computerization.

Reliability

When an analyst is faced with the tedious assignment of manipulating large quantities of data, he is often subject to that all too frequent problem of human error. Considerable amounts of time are spent, not only acquiring and manipulating data, but also rechecking the results. Computerization is one solution to this problem.

> In experimental work, automation may be necessary to collect and process large volumes of data generated efficiently and accurately.[16]

> With a computer in the experimental loop, analysis of clinical laboratory tests is both more reliable and more repeatable than is possible with manual analysis.[17]

The use of a computer not only virtually eliminates human error from the system, but may also serve to standardize the results. The results derived from an automated system are of no value if they cannot be trusted as reliable measurements. An automated system must demonstrate its reliability and repeatability before it could possibly be accepted as a laboratory tool, especially in such a field as diagnostic medicine.

> In the medical context, this is *sine qua non*. Better no no lab news at all then (*sic*) news the physician can't trust. In the absence of news, he can always find other sources of information, even his own eyes and fingers.[18]

The responsibility for reliability is not that of the automated system however. It must be remembered that, even though the analyst's job has been vastly simplified, his shoulders still must bear the burden of responsibility for accuracy.

> The ability to automate laboratory analyses is enhanced by an increasing variety of programmable laboratory instruments that can be adapted to many analytical procedures. Occasionally, some modification of method may be required. However, automation itself cannot add precision to a laboratory. Automation must be built on a sound base of reliable analytical procedures performed by trained and competent personnel.[19]

Morale

One important factor to be considered in the automation of a laboratory, as eloquently reported by Rappoport and Rappoport, is that of the morale of the technicians; especially those who feel they are being "replaced by machines".

> A good general rule in introducing automation into clinical laboratories is to start slowly. Abrupt shifts in habits are difficult to assimilate. The intelligent and safe method is to make the transition gradually. Long range goals are valuable, but it is unwise to attempt to attain them in one convulsive effort.[20]

Dr. and Mrs. Rappoport suggest allowing the technicians time to adjust to the changes in the laboratory and a chance to realize that they are still a valuable "part of the team".

> New tools that can improve and uplift the human condition now exist for hospital laboratories--the remaining task is to encourage their enthusiastic acceptance by hospital personnel.[21]

Disadvantages of Automation

Automation can be disadvantageous in given situations. For example, if the size of the hospital is so small as to warrant only a small number of laboratory tests per month, automated techniques would prove to be more expensive than conventional manual procedures. Also, a certain amount of expertise is required to do the system analysis and programming required for computer automation. If this expertise is not readily available, the additional expense of either training or hiring such a person must be considered. This, oftentimes, means an additional employee with a corresponding additional salary.

A third consideration is that the workforce of the hospital must be flexible enough in their training and skills to allow for the introduction of automated devices to the laboratory. Automation often means that the existing workforce must learn to use the new equipment or face the possibility of being "replaced by a computer".

This presents the fourth disadvantage of automation: a drop in morale. As was discussed in the above references, the decrease in morale, at the advent of automation, is a very real problem with which one must deal or face failure of the program. This failure may be due to a variety of reasons, ranging from stubborn unwillingness to learn how to operate the new equipment, to sabotage.

One must take a good look at all available and pertinent information before deciding to automate an existing system. A guideline for deciding just when automation should be considered is presented below.

When to Automate

Once the decision has been made to automate a laboratory, the decision must be made as to when to automate.

> There are distinct and quantifiable breakpoints which separate the decision not only to use an automated rather than a manual procedure, but also as to which type of automated equipment, if any should be used.[22]

Table 5[23] illustrates these breakpoints.

TABLE 5

BREAKPOINTS FOR AUTOMATION[23]

BLOOD CHEMISTRY	
Manual Procedures	0-750 tests/month
Automated Procedure No. 1	750-3000 tests/month
Automated Procedure No. 2	3,000-11,000 tests/month
Automated Procedure No. 3	over 11,000 tests/month
HEMATOLOGY	
Manual Procedures	0-1,500 tests/month
Automated Procedures No. 1	1,500-11,000 tests/month
Automated Procedure No. 2	over 11,000 tests/month
BLOOD BANK	
Manual Procedures	0-7,000 tests/month
Automated Procedure	over 7,000 tests/month

These breakpoints were determined through cost-analysis techniques and, it was found, continuing to use manual procedures, or the inappropriate automated procedure, beyond the limits of these breakpoints is costly, both in time and reagents, as well as overall expense.

Automation is not the answer to the various problems with which clinical laboratories must deal, in all cases. Rather, automation must be considered carefully for each, unique situation.

Minicomputers *vs.* Large Computers

Off-Line Computers

Many scientists use computers daily in their work and much of their research is heavily dependent on these cybernetic brains. "The computer configuration with which most scientists are familiar is the off-line system."[24] This system is most closely associated with large computers and is illustrated in Figure 1[25].

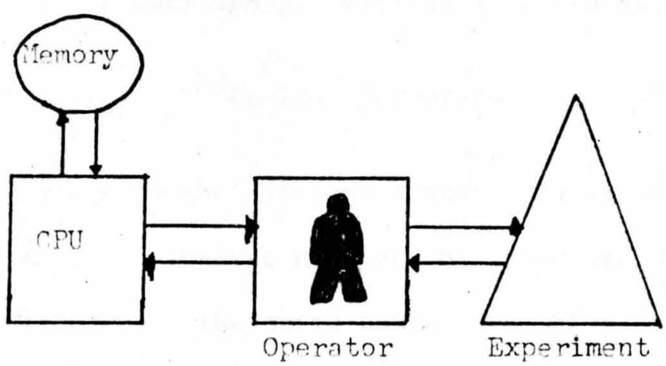


Figure 1 Off-Line Computer Configuration[25]

The description of the off-line computer system is probably best described by Perone.

> To use the computer in this configuration, the scientist typically will write a data processing program in FORTRAN or some other high-level computer language, run the experiment(s), manually tabulate the data from the strip chart recorder or oscilloscope trace, transfer the tabulated data to punched cards, add the data cards to the deck of program cards, transport the combined card deck to the computer center for processing and then wait until the program has been executed and the results printed. Turn-around times may vary from a few minutes to a few days, depending on the capacity of the computer facility, the number of users, and the backlog and priorities of work to be processed....
>
> The important common characteristics of all off-line computer systems, however, are that the experimental data are transmitted to the computer through some intermediate storage medium, the data are processed after some finite time delay has occured, and there is no direct feed back from the computer to experiment. Depending on the modes of data acquisition and transmission to the computer, the turn-around time of the computer facility, and the speed with which the investigator can interpret the result printout, the time delay for experimental modifications based on the results of previous experiments can be excessive. Should this *reaction time* be a critical factor, an off-line computer facility may be inadequate for the particular experimental studies.[26]

One will notice that there is absolutely no contact between the computer and the experiment; all communication is through the efforts of one or, usually, several human intermediaries, and the turn-around time, again depending on capacity and priorities, is often unreasonably long.

On-Line Computers

An alternative to the off-line computer is an on-line system. Using such a system, the analyst need not be concerned with the aforementioned disadvantages of the large computers. "For the investigator who requires very rapid or instantaneous results from his computer system ...the solution may be to employ an on-line computer system."[27] Figure 2 represents a block diagram for a typical on-line computer configuration.[28]

One will note immediately that, in the on-line system, the computer is directly connected to the experiment via an electronic interface. In this manner, one has all the capabilities of an off-line system as well as the following advantages.

> The most important distinction of this configuration is that there is a *direct* line of communication between the experiment and the computer. The line of communication is through an electronic interface. (This interface includes control logic, electronic elements to provide timing and synchronization, and conversion

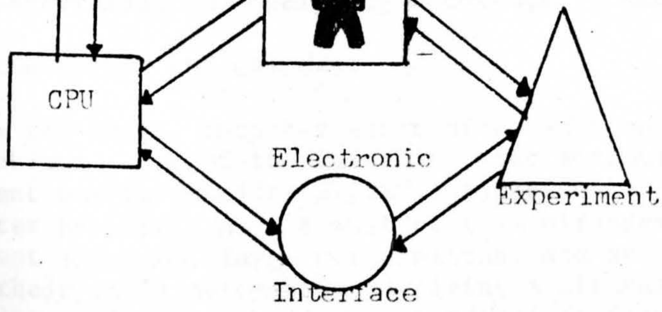


Figure 2 On-Line Computer Configuration[28]

modules--such as digitizing devices--which translate real world data into information which the digital computer recognizes.) Data are acquired under computer control or supervision, and the program for data processing is either resident in memory or immediately readable into memory to provide for very rapid completion of the computational tasks. Results may be made available to the investigator quickly by means of teletype display, line-printer, oscilloscope, or other forms of printout. In addition, the computer may be programmed to communicate directly with the experiment by controlling electronic or electro-mechanical devices--such as solid state switches, relays, stepping motors, servo-motors--or any other devices which can be activated by voltage level changes.

The advantages of on-line computer operation can be summarized as the following: *elimination of the middleman* and the concommitant substitution of an electronic interface between the computer and the experimental system, which is much more compatible with the computer's characteristics and which does not suffer from the inherent inadequacies of the human as a communication link....

The possibility of *direct computer control of the experiment* is a second advantage associated with on-line computer operation....

A third advantage...is that *real time interaction between computer and experiment* is possible. That is, because the computer can make computations and decisions at speeds exceeding most ordinary data acquisition rates, it is possible for the computer to execute experimental control modifications *before* a given experiment has reached completion....

A *breakdown of the logistic barriers of the remote computer system* is an additional advantage of on-line computer operation. Because of the direct communication link between the experiment and the computer, the mechanical and logistic road blocks typically imposed by a computer facility toward the off-line introduction of data are irrelevant.[29]

This last statement is of particular importance to scientists. The computer center, in its quest for an all-encompassing cybernetic empire, has often proved to be more of a hindrance rather than a help, and, if at all possible, should be widely circumvented in the process of purchasing, servicing, and operating a computer. Rather, there should be total user control of the computer.

From the onset, computer automation has been fraught with many problems. One of the first and most serious mistakes by management was to consider SYSTEM automation to be principally a computer problem. As a result of this misunderstanding, government agencies, large corporations, and universities placed all of their early automation involving a digital computer under the then existing branch of computational facilities. Even to this day, most computers purchased for on-line automation must have the approval of the central computing facility. Thus, earlier errors are actually being compounded. One could fill books with the case histories of failures and near failures resulting from this one simple mistake in identity.[30]

CHAPTER II

STATEMENT OF PROBLEM

Purpose

The purpose of this study is to interface the Technicon AutoAnalyzer®, an automated clinical instrument, to the Nuclear Data, ND 812 minicomputer system presently housed in the Ward-Beecher Science Building. The ND 812 minicomputer is to be used as an on-line, dedicated minicomputer.

An expanded definition of the problem follows:

1. Have the ND 812 minicomputer acquire data, via an Analog to Digital Converter, from the AutoAnalyzer, first for a blank, then for a reagent blank, followed by a series of standards, and, finally, for the samples and controls.

2. Have the standard curve calculated by the minicomputer (a Least Squares Linear Regression) and submitted for approval by the operator before continuing execution.

3. Have the computer calculate and report the values for unknown samples and controls.

4. Have a warning given if the system is out of control ($\pm 3\sigma$ on identified controls, in this case).

5. Have an alternate, back-up off-line system in case of need. Such a system will be identical to steps 1 through 4, except that the ND 812 minicomputer, specified in step 1, will be replace by a microprocessor to be used for the acquisition of data.

Rationale

Since every minicomputer is unique and different from any other minicomputer, this study is offered as merely an example of the vast capabilities of automation via minicomputers. Also, in order to develop a self-sufficient, automated laboratory, it is essential to develop the necessary abilities to operate and maintain a computer with total independence from the computer center.

> It is virtually impossible to develop automated systems for laboratory instrumentation which will satisfy all users. Thus, it becomes necessary for each of the various laboratories to develop some in-house skills which will allow the intelligent modification of existing systems of the development of entirely new systems.[31]

It is for this reason that this research has been undertaken and the results are offered as merely a stepping-stone to the further development of an independent, automated laboratory here at Youngstown State University.

CHAPTER III

THEORY OF OPERATION

Computers

Hardware

Modern computers are the natural evolutionary product of the Computers which were built during World War II. These computers were externally programmed by making manual adjustments of the connections from one unit to another. In 1947, John Von Neuman

> ...permantley wired a selection of operations for groups of units, and then placed these under central control. He suggested that numerals be treated as instruction codes, which could be stored electronically just as data numerals were stored, thus eliminating special wiring. This stored-program concept led naturally to the development of self-modifying computers since machine commands could now be manipulated by arithmetic operations.[32]

The structural design of a computer is centered around the Central Processing Unit (CPU). (Refer to Figure 3.) Each of these components will be discussed in some detail below.

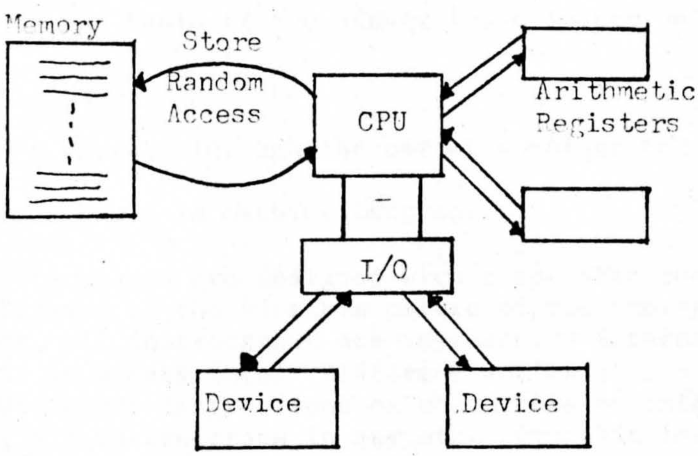


Figure 3 Digital-Computer Configuration[33]

Memory

The memory is a component capable of storing literally thousands of binary coded (digital) packets of information. Each packet is composed of *n* binary digits (bits) and is called a word of information. The ND 812 system uses a twelve-bit word, other systems may use eight, ten, or even sixteen-bit words. Each of these *n*-bit words has an address associated with it, and the information contained within can be fetched or stored randomly by the central processing unit.

Central Processing Unit

> The Central Processing Unit (CPU) is the workhorse of the computer. It controls the overall operation. Specifically, the CPU is made up of electronic registers and logic circuits which execute the simple logical and arithmetic operations of which the computer is capable.[34]

The CPU, through the marvels of modern electronics and the lightning-flash speed of electricity, can process information in microseconds (μsec) and, in larger computer systems, in nanoseconds. The CPU is comprised of a series of logic circuits that interpret electronic pulses and/or the lack of such a pulse. Hence, this "on or off" situation is the basis of the binary logic system upon which computers operate.

All CPU's operate through the use of a unique set of binary coded instructions known as Machine Language.

> Machine Languages are designed with a specific computer in mind. Because of the bistable nature of the storage unit of computers, all instructions are represented internally in the computer in binary form. Different series of 0 and 1 bits represent different instructions or characters of information. Ultimately, all instructions in any other symbolic form must be reduced to binary notation before they can be executed by the computer.[35]

When the arithmetic and/or logical operations are executed in the appropriate sequences, the computer can accomplish complex mathematical or data processing functions. Moreover, if one provides the appropriate electronic interface, these simple operations can be used to control experimental systems, acquire data, or print results via Teletype printer, line printer, oscilloscope, or other peripheral devices.

Arithmetic Registers

The arithmetic registers are high-speed electronic accumulators (AC's). Each is a set of n electronic two-state devices (like flip-flops), which can be used to accumulate intermediate results of binary arithmetic involving n-bit data. Nearly all the arithmetic and logical operations of the CPU are carried out on data contained in the arithmetic registers. Binary information can be transferred to or from memory and the arithmetic registers by the execution of appropriate machine instructions.

Control Switch Register

The control switch register and displays are used for instant communication between the operator and the computer. The switch register (SR, Appendix B) consists of an array of n switches and can be used to select an address in memory, to deposit binary instructions or data in memory or an arithmetic register, or to communicate directly with the central processor. The console display provides a binary representation (n-bit word with one light per bit) of current contents of various registers within the computer.

Input/Output

The Input/Output (I/O) bus allows transfer of binary-coded information to/from peripheral devices, such as those mentioned above.

Software

The sequence of instructions to be executed by the computer is called a program, which is actually a set of binary-coded instructions stored in memory. The CPU fetches each instruction from memory, interprets and executes it, and then moves on to the next instruction. The CPU fetches instructions sequentially from memory unless told to do otherwise by one of the instructions.

All the programs employed in computer operation can be divided into three general catagories: Developmental, Utility, and Diagnostic.

Developmental Programs

Included in developmental languages are the higher level, conversational languages such as BASIC, FORTRAN, COBOL, AND Nutran and Orcal, which are languages especially suited for the ND 812 minicomputer. User-written programs are also of this type.

The advantages of using conversational languages, are evident. The operator has complete interaction with the computer and is able to write programs that use easily understandable symbols, usually in the form of words or abbreviations, and combine any number of complex mathematical operations at one time, using one simple statement. Writing programs in high-level languages is the method of choice for the average computational problem.

The major disadvantage of conversational languages is that control of priorities, interrupts, timing sequences, flags, and peripheral devices is not possible. These functions can only be handled on the machine language--assembly language level.

Another disadvantage of conversational languages is the enormous amount of memory that is required to store the interpreter program that is used to operate the user written programs in that language. This memory space may be critical in systems with limited amounts of available core.

Utility Programs

Utility programs are usually sold with the computer and include such programs as the Compiler, General Assembler, Text Editor, and Octal Debug, and are in machine language. These programs are of great aid to the programmer in that they are used to facilitate the writing and debugging of a program as well as the translation of programs into machine language.

More specifically, a programmer will use the text editor to write and edit a program. The editor will then punch a paper tape which will be read by the assembler. The assembler will translate the program into machine language, while looking for and diagnosing errors in syntax. (Errors in logic can only be found by testing the program.) If such an error is detected, an error message is printed via the teletype. The entire program is outputted via the teletype along with the octal (base 8) code of each instruction. A binary translation is also punched out by the teletype's paper tape printer. This tape is then read into memory and the program is then executed by the computer and tested for errors in logic.

Diagnostic Programs

Diagnostic programs are used to test the logic circuits of the computer when a malfunction is suspected. If an abnormal condition is diagnosed, a message is outputted via the teletype or switch register console. By referring to the reference manuals, the operator, many times, can pinpoint the integrated circuit (IC) that is causing the trouble. It is essential that the operator have a working knowledge of the diagnostic programs supplied with the computer.

Technicon AutoAnalyzer

Hardware

The Technicon AutoAnalyzer was designed by Dr. Leonard Skeggs, an alumnus of Youngstown University. The AutoAnalyzer is used by clinical laboratories for the determination of a variety of components in human serum samples.

A schematic representation of the AutoAnalyzer is shown in Figure 4[36].

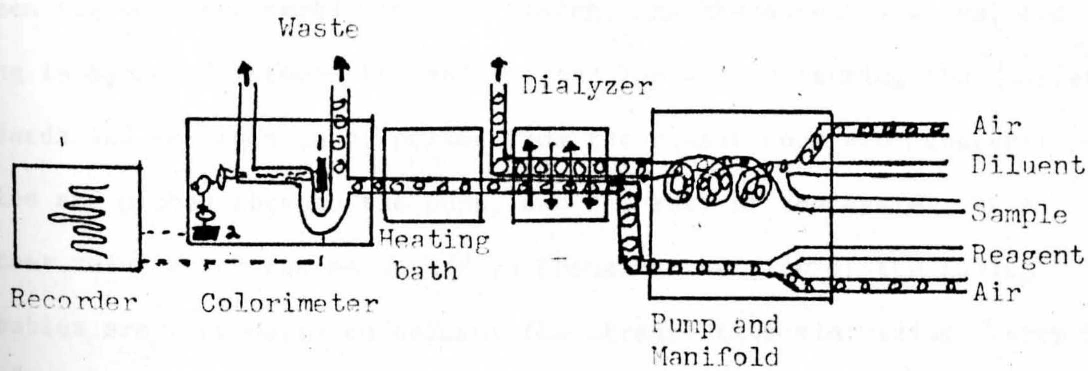


Figure 4 Schematic Representation of the AutoAnalyzer[36]

Sampler

The front end of the AutoAnalyzer consists of the sampler, which will sample up to forty serum samples in a 2:1 (sample:wash) ratio. For example, at a sampling rate of 60 samples/hr., the sampler will aspirate the sample for forty seconds and air for twenty. During the twenty second wash cycle, reagents (Ferricyanide and Saline) continue to flow, thus washing out the previous sample. Also during this twenty second wash cycle, the sample tray rotates so that the next sample is in position for aspiration.

Proportioning Pump

"The multiple proportioning pump is the heart of the AutoAnalyzer."[37] The pump consists of two, parallel stainless steel chains connected by five equally spaced rollers. These chains are gear driven which allows the entire assembly to move in an elliptical path, causing the rollers to be pressed against the rocker-type platform, or platen.

A series of flexible plastic tubes of various diameters, is placed between the roller assembly and the platen. As the assembly moves, the tubing is squeezed between the rollers and the platen causing the samples, standards and reagents to be pulled from the sample cups and reservoir bottles and pushed through the pump, and the rest of the system, at a constant rate, which can be varied by changing the size of the tubing. Air bubles are introduced to segment the stream, thus minimizing "carry over"[d] effects. The arrangement of plastic tubes, mixing coils and connectors is called the manifold.

---

[d] See Glossary.

Dialyzer

The dialyzer, through the use of a semipermeable cellophane membrane, will separate precipitates and proteins in order to obtain an interference free analysis.

The dialyzer simply consists of two, spirally grooved, matched plates separated by the membrane which generally has a pore size of 40 to 60 Å. There is also a constant temperature assembly which maintains a 37°C bath. Thus, the effect of ambient temperature changes on reproducibility is subsequently eliminated.

The sample, in a saline solution, is passed into the top plate of the dialyzer while the reagent is passed through the bottom plate. Both streams must be moving at the same rate. The pore size in the membrane is large enough to allow the glucose to pass through the membrane, by osmotic pressure, into the recipient stream, while blocking the passage of larger molecules.

Since all standards and unknowns remain in the dialyzer for the same length of time, at precisely the same temperature, and are exposed to the same amount of membrane area, the only variable to be tested is the concentration of the glucose.

Heating Bath

The heating bath is not used in all tests performed by the AutoAnalyzer but only those requiring heat for color development, enzymatic reaction, hydrolysis and digestion.

> The heating bath is a double-walled, insulated vessel in which a glass heating coil or helix is immersed in mineral oil. The mineral oil is kept at a constant temperature by a heating element regulated by a thermoregulator. The mineral oil is constantly agitated by a stirrer to ensure a uniform distribution of heat. All samples passing through the glass heating coil receives (*sic*) exactly the same temperature-time exposure.[38]

The temperature used in glucose determination is 95°C.

Colorimeter

All AutoAnalyzer colorimeters employ a dual-beam optical system. Both the reference and sample beams emanate from a single light source. The reference beam passes through a collimating lens, an aperture, and a filter before reaching its photo cell. The sample beam passes through a set of focusing mirrors, a filter, and the sample flowcell before reaching the sample photocell.

As the light beams strike the photocells, the light energy is changed to electrical energy. The voltage produced by the photocells is directly proportional to the intensity of the light striking them. The ratio of the sample to reference voltage is measured by the recorder. Thus, when the light intensity reaching the sample photocell is exactly equal to the light intensity reaching the reference photocell, the ratio is one or 100% T.[39]

The sample flowcell is connected to a debubbler device which removes the air bubbles from the flowcell before the sample reaches the optical path of the sample beam (Figure 5[40]).

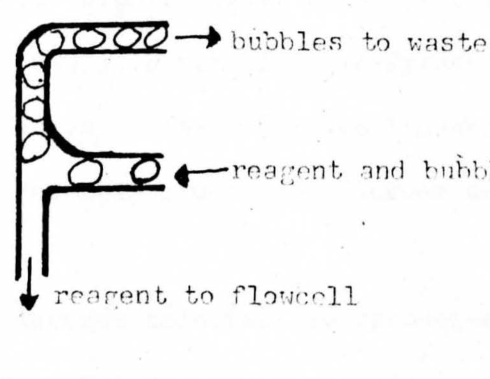


Figure 5 Debubbler[40]

Recorder

The recorder used was a d-c voltage null-balance potentiometric recorder, as manufactured by the Bristol Company of Waterbury Connecticut.

A 1-K, ten turn potentiometer was added to the bridge unit which, through the use of a constant external voltage (in this case the Voltage Reference Source), allows for the tapping of the voltage-divided, analog signal. This tapped analog signal is then to be interfaced to the ND 812 mini-computer.

Chemistry

The AutoAnalyzer has been designed to perform a variety of clinical laboratory procedures that were, formerly, very tedious when done manually. Among these procedures are the determination of: Acid Phosphatase, Alkaline Phosphatase, Serum Amylase, Albumin, Creatinine and Urea Nitrogen (simultaneously), Calcium/Inorganic Phosphate (simultaneously), Carbon Dioxide/Chloride (simultaneously), Creatinine, Chloride, Carbon Dioxide, Micro Carbon Dioxide, Serum Calcium (fluorometric), Calcium, Glucose/BUN (simultaneously), Micro GLucose, Hemoglobin, Inorganic Phosphate, Phenylamine (by fluorometer), total Protein, Serum Glutamic-Oxalacetic Transaminase (SGOT), Serum Glutamic-Pyruvate Transaminase (SGPT) (by fluorometer), Serum Glutamic-Pyruvic Transaminase (SGPT), Uric Acid, and Glucose. For this study, the Glucose methodology was chosen as one example.

Glucose is determined by inverse colorimetric techniques involving the potassium ferricyanide--potassium ferrocyanide oxidation-reduction reaction. The yellow potassium ferricyanide solution is reduced, as a result of the reaction with glucose, to the colorless ferrocyanide. The color intensity is measured at 420 mu using a flow cuvette which has a 15 mm light path.

The glucose concentration can be calculated by measuring the amount of light that is absorbed by the sample after the reaction is complete.

Calculations

According to Beer's Law,

$$A = abc, \qquad \text{(Eq. 1)}$$

where: A is the absorbance, a is a constant, b is the path length, and c is the concentration.

It becomes obvious, then, that absorbance is directly related to concentration.

In an extension of Beer's Law, we find that

$$A = -\log T \qquad \text{(Eq. 2)}$$

where: A is the absorbance and T is the transmittance.

The recorder pen traces the change in transmittance as determined by the colorimeter. The additional potentiometer, which modified the recorder, allows for a voltage-divided analog signal to be outputted from the recorder which is directly proportional to the signal which the recorder receives from the colorimeter.

In the claculation of Glucose concentrations, the %-Transmittance is read for each of the samples, controls, standards, reagent blank, and blank solutions. Formerly, this value was read directly from the chart recorder and computations were done manually.

Absorbance is then calculated by the formula

$$A = 2 - \log \%T \qquad \text{(Eq. 3)}$$

which is derived from Eq. 2. After the absorbance is calculated for each of the blanks and all standards, a linear least squares regression curve is calculated from the data points in order to determine the best straight line through those points. The slope of this line is calculated by the formula

$$a_1 = \frac{\sum xy - \frac{\sum x \sum y}{n}}{\sum x^2 - \frac{(\sum x)^2}{n}} \qquad \text{(Eq. 4)}^{41}$$

where: a_1 is the slope, x is the glucose concentration, y is the absorbance, and n is the number of standards.

The intercept is calculated through the use of the following formula:

$$a_0 = \overline{y} - a_1\overline{x} \qquad \text{(Eq. 5)}^{42}$$

where: a_0 is the intercept, a_1 is the slope, $\overline{y}$ is the average of all absorbances, and $\overline{x}$ is the average of the concentrations.

The coefficient of determination, which is used to determine the reliability of the line is determined in Equation 6,

$$r^2 = \frac{\left[\sum xy - \frac{\sum x \sum y}{n}\right]^2}{\left[\sum x^2 - \frac{(\sum x)^2}{n}\right]\left[\sum y^2 - \frac{(\sum y)^2}{n}\right]} \qquad \text{(Eq. 6)}^{43}$$

where: r^2 is the coefficient of determination, x is the glucose concentration, y is the absorbance, and n equals the number of standards.

Once the slope and intercept are calculated, within an acceptable degree of reliability, the standard curve is plotted by the technician and all subsequent glucose concentrations are obtained directly from this graph. The glucose test is a "loss of color test" meaning that the slope is a negative one and the absorbance of the reagent blank is higher than that of the samples. This process involves the calculation of absorbance for each sample, by the technician, before glucose concentration can be determined.

The entire manual procedure is both tedious and prone to error. Through the automation of this, the last step of an "automated" process, much time and money is saved and most, if not all, errors are eliminated.

CHAPTER IV

EQUIPMENT

AutoAnalyzer

In addition to the AutoAnalyzer, as previously described, the following equipment was used during the course of this research.

ND 812 Minicomputer

The ND 812 minicomputer, manufactured by Nuclear Data, Inc., Schaumburg, Illinois, 60172, was designed particularly for use in nuclear research and data processing. This minicomputer, presently housed at the Ward Beecher Science Building, is called System 4410 and is a 12-bit word computer. The system presently has 16-K of memory. (Note: 1-K of memory is equivalent to 1,024 words.)

System 4410 is a data acquisition and display system which acquires and processes data from various analog to digital converters or digital output devices. The 4-K of memory used by this system is broken down so that data may be stored in 1024 twenty-four bit words, or 2048 twelve bit words. The remaining 2-K of memory is used to store programs for handling the data. Software and hardware are provided to display or read out to any peripheral device, or process selected portions of the acquired data.

The Analog to Digital Converter (ADC), supplied with System 4410, will take an analog signal, digitize it, and present the digitized value to the computer for analysis.

> The ND560 ADC is designed to process the type of amplitude modulated signals encountered in measuring fast random phenomena. It may also be used to sample dc or slowly varying voltages.[44]

The ADC will accept a three to ten volt positive strobe pulse, one to ten microsecends in duration, via the rear-panel BNC, to open the linear gate for a pre-determined time. The external strobe rate can be up to 8,000 KHz. The internal strobe rate is 50 MHz.

A standard ASR-33 teletype is used for input by the operator and output by the computer. Through the use of the teletype, paper tapes can be read into memory or punched, by the computer, from memory, in addition to the obvious I/O capabilities of the keyboard.

Other peripheral devices can be used, such as a magnetic tape reader.

Analog - Digital Designer

> The Analog-Digital Designer (ADD) is part of the Heath/Malmstadt-Enke Modular Digital System. It is an exceptionally versitile device for teaching digital logic and instrumentation, and for experimentation, research, and development as a permanent or semi-permanent instrument in particular configurations.[45]

The ADD is comprised of three modules: the Power Supply, the Binary Information Module, and the Digital Timing Module.

Using the dual monostable card in conjunction with the Digital Timing Module, an 8.6 μsec pulse of 3.6 volts can be obtained. This pulse is used as an external strobe for the ND 812 ADC, which initiates the ADC process. The duration of the pulse was determined by the programmable timer and the amplitude by an oscilloscopic measurement.

By connecting the Digital Timing Module to the Binary Information Module, one can cause a lamp, on the Binary Information Module, to light at the beginning of each timing sequence, thus indicating to the operator that a pulse has been generated. The power for these two modules is supplied by the third module, the Power Supply Module.

The delay between pulses can be adjusted using the Multiplier Switch, Variable Calibration (Var-Cal) Switch, and Variable Control Knob on the Digital Timing Module of the ADD, so that the frequency of data acquisition can be changed as needed for a given experiment. For this study, a one-second pulse rate was chosen.

Programmable Timer

The programmable timer used was a Heath-Schlumberger Programmable Timer, Model Number SM-102A.

> This Programmable Timer is a compact, lightweight, electronic timing instrument capable of 100 ns resolution (direct count). Both Start and Stop inputs are internally switch-selected to allow for either a zero crossing or TTL level signal. Although precise Time A-B measurement is the primary function of the timer, it will also measure and display Period, Events/Scaled Events, Frequency Ratio, and Period Average.
>
> Display for this instrument consists of five 7-segment LED (light emitting diode) arrays, while three lamps and two decimal points provide range information. A rear panel connector provides BCD (binary coded decimal) information of the readout and allows remote programming of the instruments' seven ranges.[46]

This timer was used to determine the duration of the strobe pulse as outputted by the ADD. A one to ten μsec pulse will be accepted by the ADC. The ADD provides a pulse of 8.6 μsec.

Voltage Reference Source

> The Heath Model EU-80A Voltage REference Source is a very accurate, regulated variable DC voltage standard designed for maximum convenience and versatility. The pushbutton Function switches select the mode operation permitting the Voltage Reference Source to be used as a DC voltage standard; a 60 Hz signal source for oscilloscope calibration; or a precision DC sum or difference source for voltage comparisons, potentiometric measurements, and bucking voltage applications.[47]

The Voltage Reference Source (VRS) was used in testing the acquire routines to prove that the ratio of digitized values for the analog signals were indeed proportional to the ratio of the voltages.

The VRS was also used in trial runs to simulate peaks to determine whether or not the peak-picking subroutine was working properly.

Wave Generator

The Wavetek Wave Generator was also used in the simulations.

> The Model 142 VF VCG Generator provides sine, square, triangle, positive pulse, and negative pulse outputs (with a separate sync output) over a 0.0005 Hz to 10 MHz frequency range. Frequency range selection is provided in ten decades; and a vernier control permits adjustment to within approximately 1% of the range selected....
>
> With this instrument it is possible to simultaneously program and sweep the output frequency, select the output symmetry desired, and manually or electronically vary the dc offset. This capability, coupled with the variety of waveforms available and precision output amplitude control makes the Model 142 an extremely versatile instrument for ...laboratory applications.[48]

The wave generator was used to provide a slowly varying dc signal (sine wave of 1-5 volts) for the ADC during the simulation runs.

Voltmeter

The Simpson, Model 250, Voltmeter was used to check the output of the Wavetek Wave Generator. The ADC will accept a 0 to +8 V nominal signal input.

E & L Micro-Designer

The E & L Micro-Designer™ is a microprocessor system based on the Intel 8080 chip. This chip is used for processing data and control; it is well documented and a great deal of software has been developed for use with 8080 systems. It can perform an additional 78 basic functions; more if the various combinations are considered. Operations include data transfer, logical and math operations, input and output of data, decision making and branching.

The basic system is composed of four plug-in cards, the control panel, the interface board, plus the power supply and software.

The Micro-Designer has been interfaced to a modified ASR-33 teletype which greatly facilitates the programming and allows for hard copy documentation of programs.

The Micro-Designer was used to create a data tape for use as simulated data to test the final program. This tape was generated through the implementation of the D-Bug program; a software debugging and data entry package furnished by Tychon, Inc.

Through the use of this program, the operator may enter into memory specific data at specified addresses. Finally, a paper tape may be punched containing these data.

The paper tape, punched by the Micro-Designer, has a two-byte word format, the first byte consisting of two bits (the most significant bits (MSB)); the second byte consists of six bits (the least significant bits (LSB)).

This tape is compatible with the ND 812 in that the ND 812 will read paper tape in two-byte words also, each byte being six bits in length. Since the Micro-Designer uses an eight-bit word, and the ND 812 uses a twelve-bit word, the four MSB's of the ND 812 word will be zero by default. This in no way changes the reliability of the data as this fact was taken into account when the data tape was generated.

The data tape was then used in place of the actual AutoAnalyzer data for "dry run", or simulation, testing.

CHAPTER V

LANGUAGES AND PROGRAMS

Machine Language

Machine language is the language to which, ultimately, all computer programs must be reduced. Machine language is unique for each type of computer as it is based on the logic circuits of the hardware. Since these logic circuits are a series of bistable (on-off) components, machine language is of binary format.

The ND 812 uses a twelve-bit binary word, and all machine language instructions for the ND 812 minicomputer system can be any number from 000000000000_2[e] to 111111111111_2, inclusive. The complexity of these numbers becomes obvious when trying to keep track of twelve digit numbers comprised entirely of ones and zeros. Fortunately a simpler system has been devised.

The Octal (base eight) System of assigning numerical values to binary forms is useful as a shorthand method of writing pure binary numbers. This system deals with groups of three binary digits such that, in any octal digit, only eight possible combinations of binary positions occur (that is, 000, 001, 010, 011, 100, 101, 110, and 111). The octal equivalents of these representations are: 0, 1, 2, 3, 4, 5, 6, and 7, respectively.

---

[e]Subscript 2 indicates base two, or binary.

Therefore, the twelve-bit ND 812 word can be written as a four digit octal word, which is easily transcribed into or from its binary equivalent as well as its decimal equivalent.

The decimal equivalent is determined by multiplying each digit by 8^n, where n is equal to the position the digit holds, and adding the products. For example,

145_8[f] $= (1x8^2) + (4x8^1) + (5x8^0) =$ 101_{10}[g] .

And, as a further example,

$$001100_2 = 14_8 = 12_{10} .$$

Computer Word Formats

Storage Data Word Format

The basic data word format for the ND 812 is shown in Figure 6[49].

Since the ND 812 is oriented towards 12-bit words and the octal numbering system is employed, the value representable in any single word will range for 0000_8 to 7777_8, or from 0_{10} to 4095_{10} representing 4096_{10}

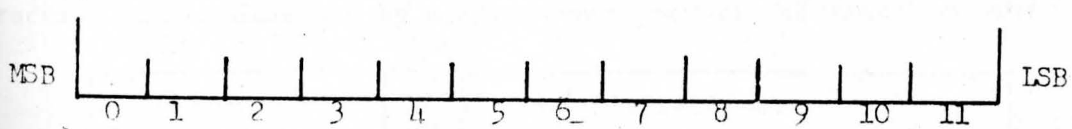


Figure 6 Data Word Format[49]

---

[f] 145 to base 8

[g] 101 to base 10

possible values. This is precisely the same number of words in a standard ND 812 memory stack, thus, a value contained in a single 12-bit word can address any location within the stack.

The leftmost bit (bit 0) is the most significant bit (MSB) and the rightmost bit (bit 11) is the least significant bit (LSB).

Two's complement arithmetic[h] is employed in the addition and subtraction operations of the ND 812. Bit 0 may be used to test the polarity of the number. If bit 0 equals 0, the number is positive. If bit 0 equals 1, the number is negative.

Instruction Word Format

Single-Word Format

Single-word memory reference instructions are of the format shown in Figure 7[50]. Single-word memory reference instructions occupy only one 12-bit word. The six address bits can specify a displacement which is added to the program counter to obtain the effective address. Because the range is from 0 to 63_{10}, that is the range of addresses which can be accessed. Bit 5, however, can specify whether this range is forward or backward, so that any data word that is within $\pm$ 63 locations of the instruction can be accessed by a single-word memory reference command.

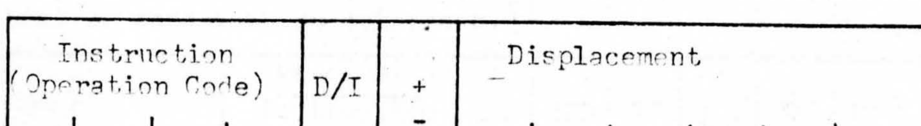


Figure 7 Single-Word Format[50]

---

[h]See Glossary.

If bit 4 is set to 1, indirect addressing is permitted. This means that the contents of the location which is ± 63 locations from the instruction location is used as a pointer to the actual operand.

Two-Word Format

Two-word memory reference instructions have the operation code in the first word and the absolute 12-bit address in the second. The two words must be contiguous and in the same field. The format of a two-word format is shown in Figure 8[51].

This format provides the ability to address operands in fields other than the one in which the operation resides. This is done by setting bits 10 and 11 for any of the four fields by setting them from 00 to 11_2. Setting bit 9 to zero cancels this effect.

Bit 8 determines which of the main accumulators are to be used, and bit 7 allows the selection of an indirect address.

Literal Format

The format for a literal instruction is shown in Figure 9[52].

These instructions permit the programmer to save both time and storage space, because the literal instructions enable the storage of counter initialization constants, increment and decrement constants, and logical AND masks in the instruction which uses the value. This saves space otherwise needed to store the constants seperately and the time to access these constants.[53]

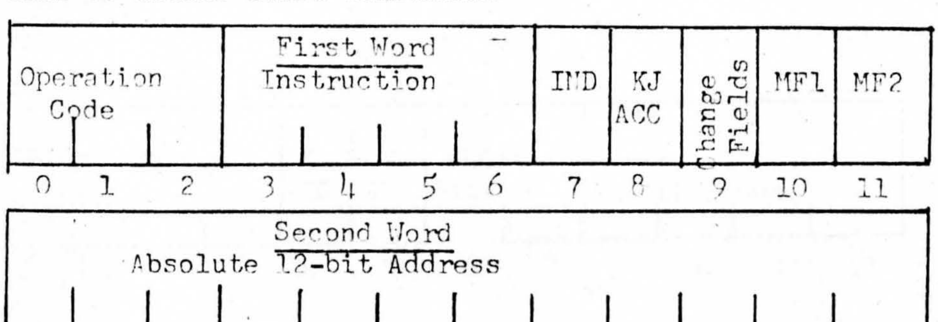


Figure 8 Two-Word Format[51]

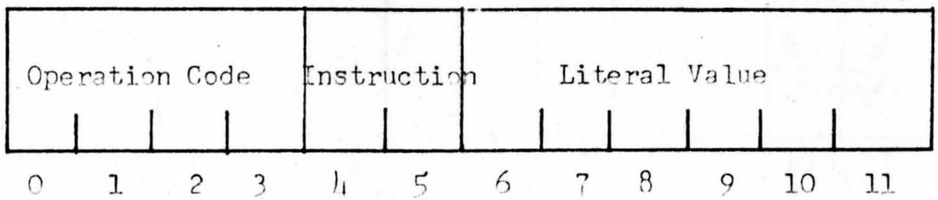


Figure 9 Literal Format[52]

Group 1 Instruction Format

All instructions of the Group 1 type have the characteristic bit pattern 0010 in bits 0-3, inclusive. These instructions are generally of the arithmetic, logical, exchange and shifting functions in operations on the internal accumulator registers. Group 1 instruction format is shown in Figure 10[54]. This group also contains the hardware multiply and divide instructions.

Group 2 Instruction Format

Group 2 instructions are primarily concerned with testing for internal conditions of the main accumulators. Several variants of these instructions can also test, set, clear and complement the overflow and flag bits; others can complement, increment, and negate the contents of the J and K registers. The format for these instructions is illustrated in Figure 11[55].

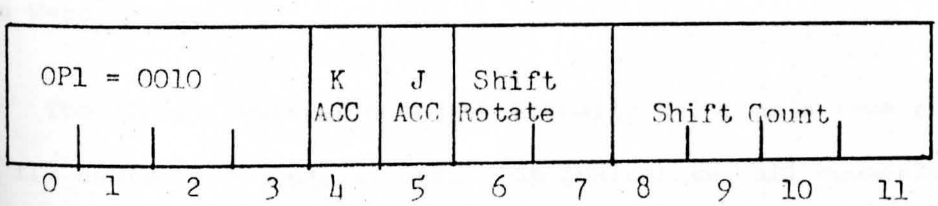


Figure 10 Group 1 Instruction Format[54]

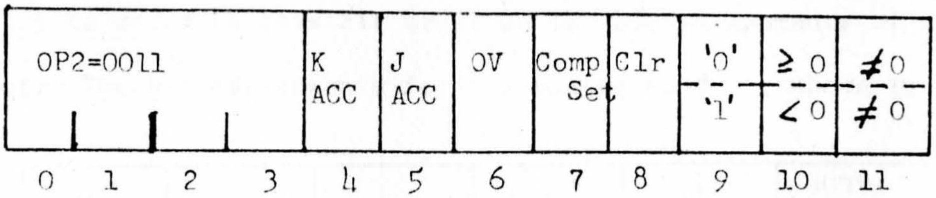


Figure 11 Group 2 Instruction Format[55]

The instructions in this group are microprogrammable, i.e. they can be OR'ed together to produce both results. The bit pattern constituting the instruction may be combined to produce different effects.

For example, the instruction code for CLEAR is 1410; the instruction for CMP (Complement) is 1420. These two instructions cause the flag to be cleared and then complemented (set equal to 1). If these instructions are OR'ed in their binary format, the command becomes:

	001100001000	(CLEAR)
OR	001100010000	(CMP)
which equals:	001100011000	(SET), or 1430.

This command will unconditionally set the flag equal to 1.

When a condition is tested via the group 2 instructions, the ND 812 takes one of two possible actions:

1. If the condition is true, the next instruction is skipped over;
2. If the condition is false, the next sequential instruction will be executed.

Status Word Format

The status register does not actually exist as a true register. It is the contents of several groups of indicators, all commonly accessed by storing them in the J register, when desired. Since each bit of each indicator is stored at a particular bit position of the J register, it is customary to refer to this bit order as the bit assignments of the status register. The bit assignments for the status word are shown in Figure 12[56].

Flag	OV	JPS		INT		IONL	IONA	IONR	IONH	Current Execution	
		MFØ	MF1	MFØ	MF1					MFØ	MF1
0	1	2	3	4	5	6	7	8	9	10	11

Figure 12 Status Word Bit Assignments (J Register)[56]

Machine language programming is, at best, a very tedious method of programming as the programmer must keep track of all operands, addresses and counters while programming. Also, the instructions have to be toggled in (keyed in) manually by means of the switch register, unless the Octal Debug Program is used which enables the operator to input four digit octal numbers via the teletype.

Assembly Language

Assembly language, while more difficult to use than the conversational languages, is much less complex than machine language. Assembly languages, like machine languages, are unique to the system for which they were designed and use a set of mnemonics in place of the binary instructions which are necessary in machine language.

These mnemonics are translated into machine language by a software package, developed for the computer, called the Assembler. To aid in program writing, another software package called the Text Editor is used.

The mnemonics are extremely beneficial to the programmer in that they are much easier to remember than a set of numbers. For example, the machine language instructions for jump unconditionally, jump to subroutine, idle, stop and load memory into J are: 6000, 6400, 1400, 0000, and 5000 (all in octal notation), respectively. The assembly language instructions for these same instructions are; JMP, JPS, IDLE, STOP, and LDJ, respectively.

Notice that the mnemonics are, usually, some form of literal abreviation of the command, thus making them easier to remember than a set of numbers.

Assembly language programming is tedious, though not quite so arduous as machine language programming. The programmer need not remember the absolute address of all variables, storage locations, and data points, as they can be assigned identifying names (symbolic addresses) and the assembler will keep track of their locations, thus freeing the programmer from this responsibility. Perhaps an example of programming would be beneficial.

A > B

The first step in writing any program is to define the problem. In this example, the problem is as follows: Input two numbers defined as "A" and "B"; compare the two numbers and determine which is larger, and output a literal statement "A > B", or "B > A", as applicable.

The second step is to write a flowchart which depicts the problem and solution in a step-by-step manner. The flowchart for this algorithm[57] appears in Listing 1, Appendix A. (All subsequent listings also appear in Appendix A.)

Third step is to translate the algorithm into a language that the computer will understand (in this case, assembly language). The assembly language program for "A > B, B > A" appears in Listing 2[58].

After this program is checked for literal errors (errors in syntax) and edited, the text editor punches a paper tape, called the source tape, of the program. This tape is then read by the assembler which will generate a paper tape on which is punched the binary machine language instructions (object tape) and type out a listing of the program in assembly language with the corresponding machine language instructions for each command. The listing of this problem, as printed by the Assembler, appears in Listing 3[59].

At the end of the third pass of the assembler, a list of all symbolic addresses and their locations is typed and this list is appended to the end of the program listing. This symbol table list is also used as a debugging aid to enable the programmer to locate, more rapidly, a symbol he may wish to check.

Also, during the assembly process, if any errors in syntax are detected, an error diagnostic will be outputted by the assembler via the teletype.

Finally, the object tape is read into memory and tested. The results of the run of this example appear in Listing 4.

A closer examination of Listing 3 shows six columns of information. The first column contains the sequential addresses of all instructions. Notice that all numbers are in octal notation.

The second column contains the octal representation of the machine language instruction for the assembly language mnemonics which are found in columns three, four and five. Column three contains the relative and symbolic addresses of variables or subroutines used in the program. They are identified by the comma which follows them. Column six contains messages, which must be preceeded by a slash (/), for the programmer's information only. They are not read by the assembler and are, in no way, essential to the operation of the program. Used merely as programming aids, these comments are used by the programmer to help to explain various steps and/or commands he/she wishes to clarify for future reference.

In the first five lines of the program, the computer is instructed to jump to the input subroutine twice to get two numbers, which are input at the teletype by the operator. In the input subroutine, as in all subroutines, the entry point must be zero. When the CPU jumps to the subroutine, the address from which it jumped is incremented and stored at the entry point of the subroutine. At the end of the subroutine, the JMP@ command causes a return to the address which is stored at the entry point. In this manner, the usually sequential operation of a program may be altered.

After returning from the subroutine, the value in the J register (the number entered by the operator) is stored in the location designated by "A" (or "B", the second time).

In the next section of the program, lines 0205 through 0213, the value stored at B is subtracted from the value stored at A. The J register is then tested for a positive or negative value. (Bit zero is zero, if positive, or one, if negative.) Depending on the result of this test (SIP J), the appropriate output message is loaded into memory and the program will jump to the output routine.

The output routine (0235-0253) will output one character at a time, decrementing a counter between each character. When the counter equals zero, the entire message has been printed, and the program will return from the subroutine and execute the statement following the JPS command, in this case STOP, which halts execution of the program. By depressing the continue key on the front panel of the computer, the next statement will be executed which is a jump back to start, and the program will run again.

Output

A second program, shown in Listing 5, was written to gain a further understanding of the output process. The length of this program, as compared to the length of the message it actually outputs (Listing 6), is obviously long. This same program could be written in two lines of a high-level language such as Nutran--

```
5 PRINT 'RET. TIME (SECS) PK. HT.  PK. AREA',/,/,'E'
10 STOP
```

--and thus save the programmer a lot of time.

Multiply

A third program was written to check the hardware multiplication function. This program and its trial run appear in Listings 7 and 8 respectively.

Once again, note the length of the output routine as compared to that of the total program (Listing 7). Also, lines 0111 through 0113 are necessary for the total multiplication process.

In Nutran, this entire program could be written in three lines.

```
5 INPUT X,Y
10 PRINT X,'*',Y,'=',X*Y
15 STOP
```

Notice that multiplication in assembly language is only effective with one digit multiplicands which result in one digit products (Listing 8). It was at this point the decision was made to use Nutran to do the data manipulation and message output.

> ...Assembly-language programming for sophisticated data processing or for formating of typewritten report generation is extremely awkward and tedious. Thus, it would appear advantageous if the best features of high-level languages and assembly language could be combined. Thus, a seemingly ideal language for a laboratory application would be one where the data processing and standard I/O could be handled by high level instructions; handling functions could be handled by assembly-language program segments.
>
> Combining high-level and low level language program segments is a perfectly feasible approach.[60]

Thus, the decision was made to do just that; acquire the data with an assembly language program and manipulate these data with a program written in Nutran.

Acquire

A preliminary acquire routine was written in order to gain familiarity with the operation of the ADC. The assembled listing of this program appears in Listing 9.

The first section of this program (lines 0001 through 0050) will analyze all interrupts received by the computer after storing the contents of all registers. If the interrupt signal is not recognized as one generated by any of the periphery or other known sources, execution will stop at line 0020.

If the interrupt signal is from the ADC or any other recognized device, the registers are restored and execution will continue.

Lines 0060 through 0070 contain the ADON subroutine which is a subroutine which will turn on the ADC and enable all high level interrupts. The ADC and high level interrupts are disabled in lines 0071 through 0077.

Upon recognizing an interrupt from the ADC, the computer will immediately trap to location 0101 in Field 0. This is a hardware trap address which cannot be over-ridden by software programs. Therefore, the ADC routine must be written in Field 0. This subroutine (lines 0101 through 0131) will first store the contents of all registers, read the ADC word (7527_8--line 0110) and store the digitized value at locations High and Low. The registers are restored and execution continues.

The SIP K command (line 0111) assures storage of a digitized value only if one has been acquired. If not, the commands to store the data are jumped over and the registers are restored, thus allowing execution to continue.

This program is started at line 0200 and immediately jumps to the ADON subroutine. Upon its return, it will sit in the idle loop to await a signal from the ADC, following which it is expected to jump to the ADOFF subroutine and stop.

In practice, however, it was found that the program, though it would acquire data, could not escape from the infinite idle loop. When the registers were restored at the end of the ADC routine, the program would simply jump back from where it came, which was the idle loop. Corrections for this problem were provided in the next program.

AutoAnalyzer Acquire Routine

The program which was eventually to be the final acquire program, is shown in a flowchart in Listing 10; the program itself is given in Listing 11. This program is a modification of the acquire routine written previously, with several linkage changes and the addition of several subroutines.

Lines 0001 through 0133 (Listing 11) serve the same purpose as in the acquire routine program, with one difference. In lines 0116 and 0117, the commands TWJPS PKPKR cause the program to jump out of the ADC routine prior to its return to the idle loop (lines 0214 and 0215). In the PKPKR subroutine (lines 0222 through 0271), the former ADOFF subroutine is used to disable the ADC and all other interrupts.

The counter (line 0230) is decremented and, on the first pass through this subroutine, a jump to STNDRD occurs.

STNDRD is a subroutine which will store the first data point as 100% T. Notice that the command 0566 (line 0274) is a machine language command for a two-word store J in Field 2, at a location represented by the contents stored at ADDRS (TWSTJ@ ADDRS). The value at ADDRS is then incremented (line 0276) and the execution will stop. (This command was written expressly for this program and is an example of the group 1 instructions as discussed on page 42 and two-word instructions as discussed on page 41.).

When the operator is satisfied that the data point has been stored satisfactorily, the program is allowed to continue by depressing the CONT key on the front console of the computer (Appendix B).

The program will then jump to ADON, return from the PKPKR routine to the ADC routine, and, ultimately, back to the idle loop to wait for subsequent interrupts.

On the second pass through the PKPKR routine, the above process will be repeated, with the datum stored in the next sequential memory location of Field 2. This is the reagent baseline (zero Glucose concentration).

On third and all subsequent passes through the PKPKR subroutine, an unconditional jump to PKS occurs (line 0234). In this part of the subroutine, consecutive data points are tested for increasing or decreasing values. If the values are increasing, the new value is stored and the old value is erased. The ADC is re-enabled, and a return to ADC--return to idle loop occurs.

If the values are decreasing, a flag is checked to see if a peak has been recognized. If the flag is not set, a jump to STPK (lines 0304-0325) occurs. In this subroutine, the datum is stored, ADDRS is incremented, and a counter, PKCTR (line 0325), which keeps a tally of the number of peaks remaining, is decremented (line 0312). If this counter is zero, execution stops. Otherwise, the flag is set and a return is generated and the process starts over again.

If the flag was set, the datum is stored, replacing the previous datum and the program returns for the next data point. As soon as the PKPKR subroutine recognizes increasing values, the flag is cleared and the above procedure is repeated.

As was mentioned above, as soon as all 40 peaks have been stored, execution will stop. Depressing the CONT key will reinitiate the program and the next set of data can be acquired for another AutoAnalyzer run.

It is imperative that the programmer have complete comprehension of machine language techniques, as it may become necessary to write an instruction for which there is no assembly language mnemonic. Refer to addres 0274 and address 0306 of Listing 11. The instruction, as previously mentioned, means, literally, to store the contents of the J register, in Field 2, at the location specified by the contents at location ADDRS. A mnemonic for this would be TWSTJ@, however, a change of field is necessary since ADDRS is a defined location in Field 0, and the data must be stored in Field 2.

Two methods may be used. The first is to program that one instruction in machine language. The second is to use the mnemonic TWSTJ@ and define ADDRS as a location in Field 2. The former method was chosen to illustrate the need for comprehension of machine language for use in situations where the programmer wishes the computer to execute an instruction for which no mnemonic exists.

The AutoAnalyzer Acquire Routine will read data and store it in Field 2 in sequential memory locations (note the TWSTJ@ followed by ISZ ADDRS commands (lines 0274-0275 and 0306-0307). Once all the data have been read and the values for the peaks have been stored in memory, a Nutran program is used to manipulate and process these data.

The loading and initialization procedure for assembly-language programs is given in Appendix C.

Nutran

Nutran is a high-level, conversational language that was derived from FORTRAN, a highly used and widely recognized scientific programming language.

Nutran resides in the first two fields of memory (Figure 13[61]) and must be loaded after the AutoAnalyzer Data Handler Program has finished acquiring data. Problems in convenience arise due to the fact that both programs must reside in the same portion of memory and each of these programs are mutually exclusive; they cannot both be loaded into the same memory field at the same time.

It is possible to load Nutran in other fields, with the use of the source tape for the Nutran Interpreter. Modifications of the source tape will cause it to be loaded into memory fields chosen by the programmer. Such a tape is not part of the software package as received from Nuclear Data which results in the fact that both programs are required to occupy the same fields of memory and, consequently, must be loaded into memory one at a time--a rather time-consuming process.

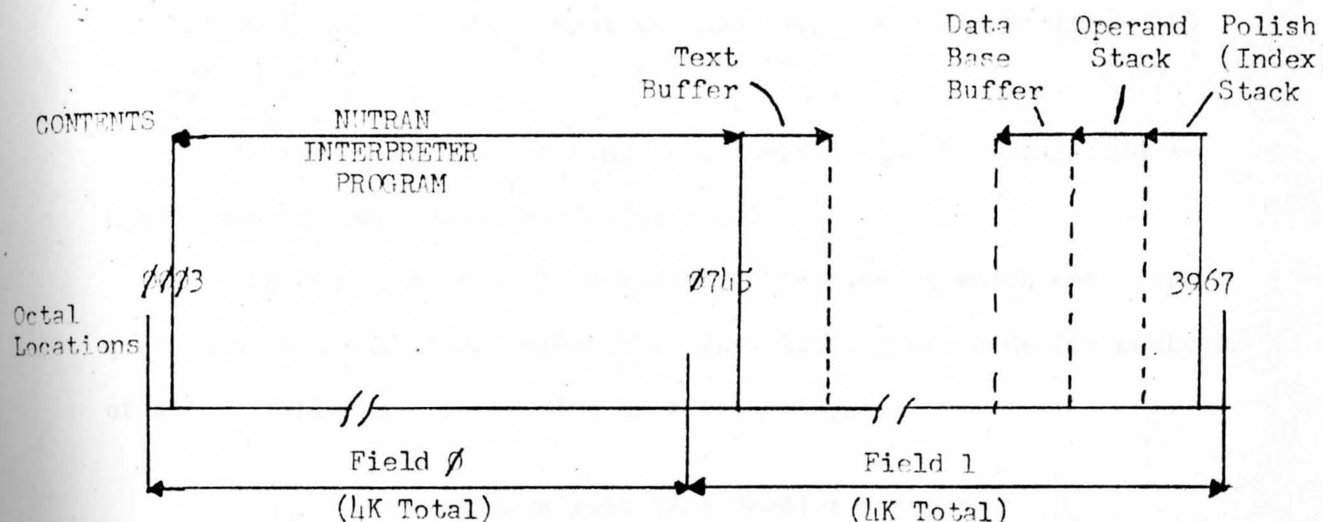


Figure 13 Nutran Interpreter, Core Map[61]

Linear Least Squares Program

The flowchart of the Linear Least Squares program appears in Listing 12. The actual program and example run are shown in Listings 13 and 14, respectively. This program will calculate the linear least squares regression constants for a given set of data points (using AutoAnalyzer data) and then calculate, if the operator so chooses, the Glucose concentrations for any number of samples. Notice that all data is supplied by the operator; this is purely an offline program.

Lines 5 through 110 handle the input of data and the preliminary data processing. After all data points have been entered by the operator, the linear least squares regression values are calculated in lines 115 through 140. These numbers are then outputted via the commands in lines 150, 155 and 160.

The operator is asked whether or not execution is to continue. This option allows the operator to decide whether or not to continue calculations or abort the run, should the standard curve be unsatisfactory. If execution is to continue, lines 200 through 235 handle the inputs of % T for each sample. The output messages are contained in lines 240 through 270.

In the trial run (Listing 14), the data can be identified as those numbers immediately following a colon (:).

Listing 15 represents a symbol table listing which can be requested at the end of an execution. This listing contains the contents of all variables as of the time execution stopped.

AutoAnalyzer Data Handler Program

The AutoAnalyzer Data Handler program, depicted in Listing 17 (flowchart in Listing 16), is a modification of the previous program. This program will operate on 41 internally stored data points (listing 18) rather than having them inputted by the operator. This program is designed for on-line operation.

The first thirty lines of the program provide instructions if the operator so chooses. Notice that the Glucose concentrations are internally defined (lines 60 through 85) and the input is now handled by the GET (I) command, which will retrieve data from memory Field 2.

Calculation will continue as before, with the regression constants being printed and providing the operator with the opportunity to continue or abort execution.

If execution continues, the operator must input the limits of the control value; the values will be printed, and an error message is typed if the control value is outside of these defined limits.

An example run is shown in Listing 19 and the symbol table is presented in Listing 20.

The loading and initialization procedure for Nutran programs and the Nutran Interpreter is given in Appendix D.

CHAPTER VI

RESULTS

Off-line Program

The Nutran program shown in Listing 13 was used as a preliminary, trial program which ultimately gave rise to the AutoAnalyzer Data Handler program (Listing 17). Its purpose was then extended to be an off-line type program, to be used in situations where direct access to the computer, at the time of the AutoAnalyzer run, is not possible. All data are input by the operator as read directly from the graph on the recorder. A typical chart is shown in Figure 14.

One important thing to keep in mind is that the baseline concentration must be a very small number (i.e. 1.0×10^{-7}) as the calculations will not support a concentration of zero.

A simulation run was performed by using actual data as supplied by the standards and simulating data for the thirty-five remaining samples. The data for this simulation appear in Table 6 and the standard curve of absorbance vs. concentration is shown in Figure 15.

It should be pointed out her that Table 6 contains % T as determined in an actual, manual run of the AutoAnalyzer, as is the case of the six standards, or as arbitrarily assigned to simulate actual samples, as was done for the remaining thirty-five data points. All Glucose concentrations, aside from the five standards, were calculated, as were all absorbances. The actual output of this off-line simulation, is presented in Listing 21, with the resulting symbol table in Listing 22.

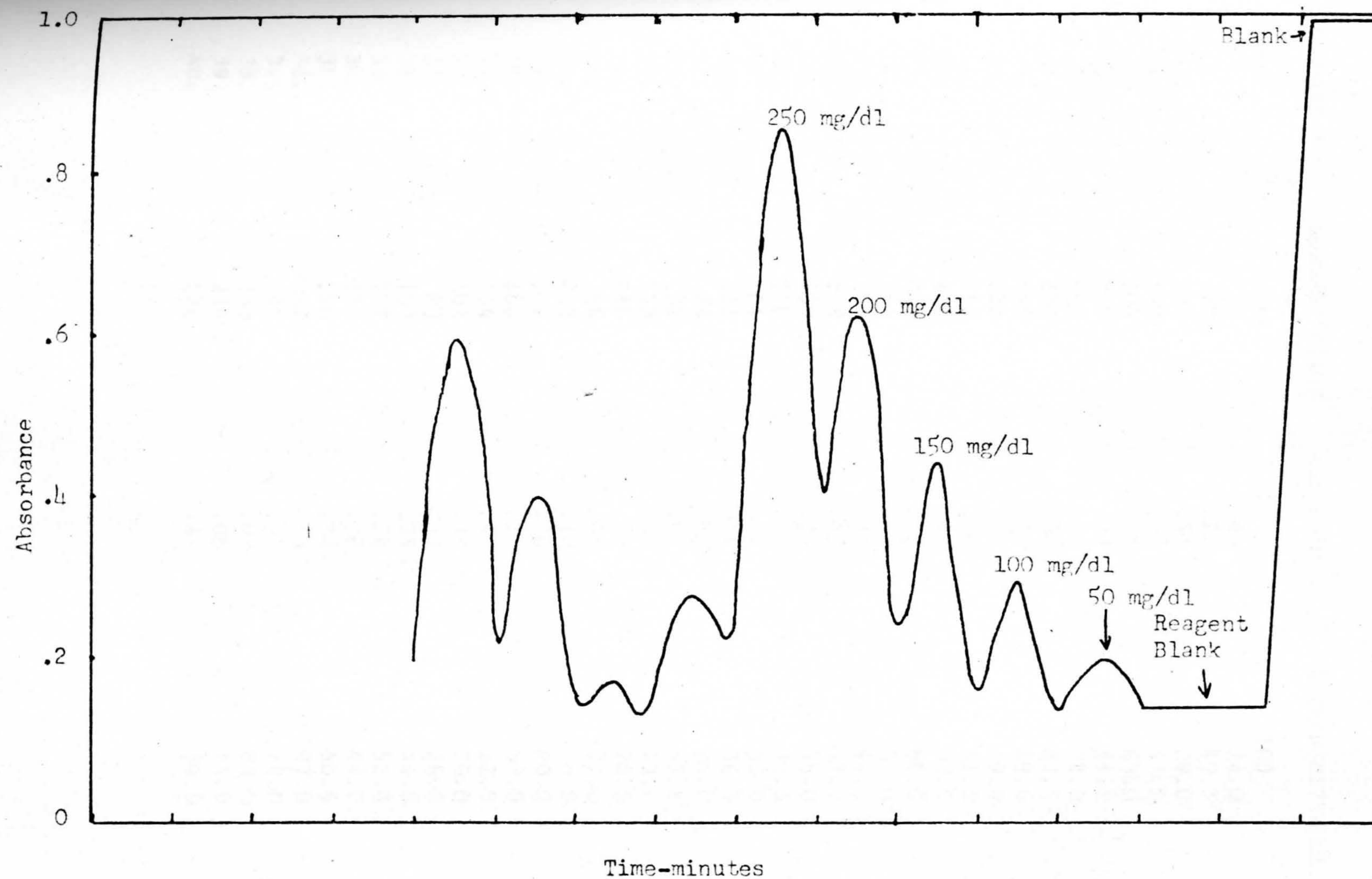


Figure 14 Graph From AutoAnalyzer Recorder

TABLE 6

DATA FOR SIMULATION

STANDARD #	CONCENTRATION mg/dl	ABSORBANCE	% TRANSMITTANCE
----	-----	0	100
1.	0	.85	14.0
2.	50	.71	19.5
3.	100	.54	29.0
4.	150	.36	43.5
5.	200	.20	63.0
6.	250	.07	85.5
7.	189	.24	58.0
8.	238	.08	82.5
9.	156	.31	49.0
10.	122	.31	49.0
11.	110	.50	31.5
12.	116	.48	33.0
13.	210	.17	68.0
14.	134	.42	38.0
15.	156	.35	44.3
16.	133	.43	37.2
17.	189	.24	58.0
18.	216	.15	71.5
19.	104	.52	30.0
20.	128	.44	36.5
21.	74	.62	24.0
22.	132	.43	37.5
23.	153	.36	43.7
24.	216	.15	70.5
25.	68	.64	23.0
26.	232	.10	78.5
27.	232	.10	80.0
28.	188	.24	57.0
29.	114	.49	32.0
30.	104	.52	30.0
31.	83	.59	26.0
32.	153	.36	44.0
33.	177	.28	52.0
34.	153	.36	44.0
35.	196	.22	60.5
36.	132	.43	37.0
37.	95	.55	28.0
38.	238	.08	83.0
39.	238	.08	84.0
40.	120	.47	34.0

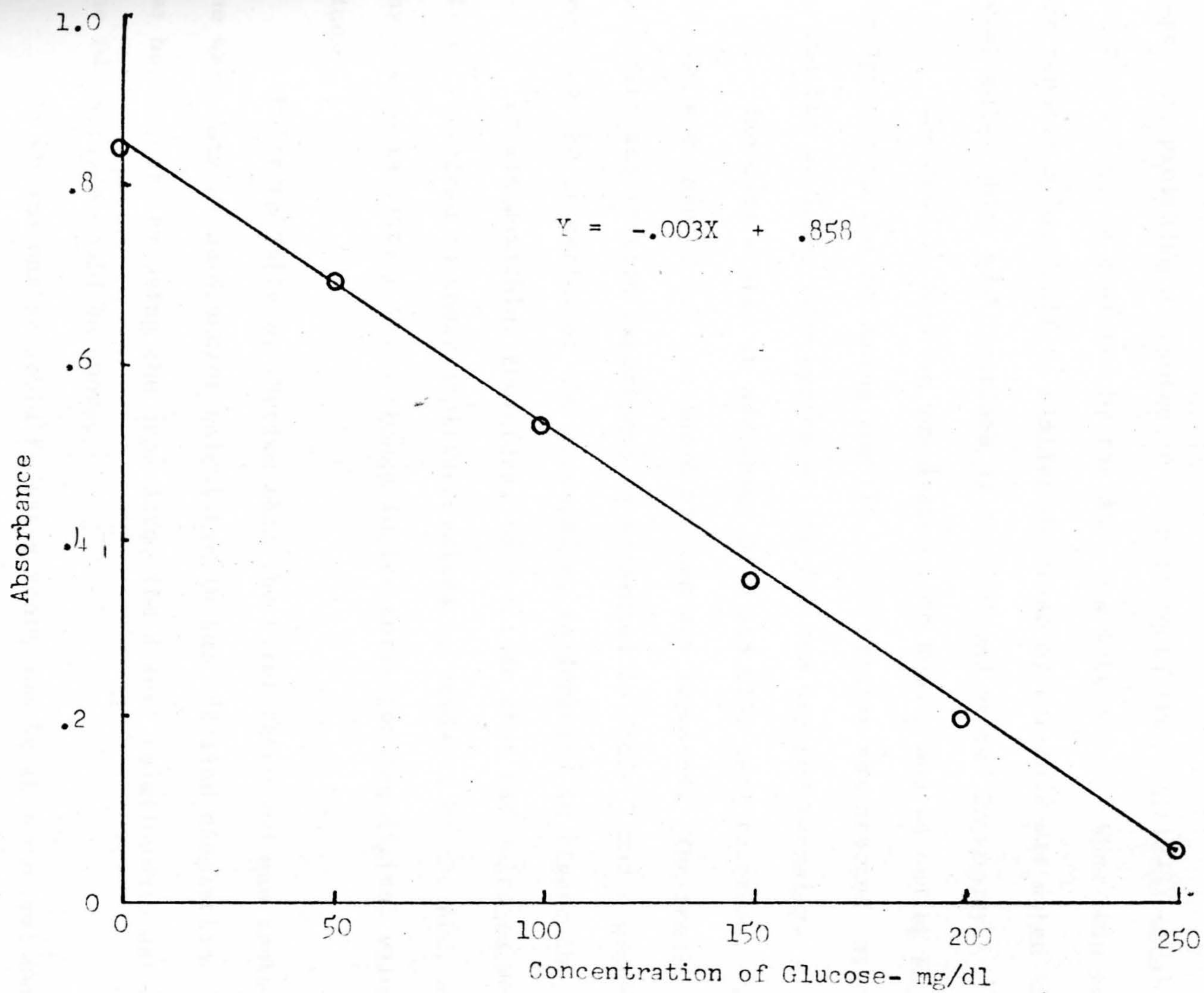


Figure 15 Standard Curve

On-Line Programs

The Acquire Routine (Listing 9), though not operative enough for ultimate use, did function as a reference source for the simulated data tape. By providing a constant 6 Volt current, the digitized, octal value for six volts, as assigned by the ADC, was determined. When this voltage was reduced by one-half, a similar decrease by one-half was noted in the octal value which was digitized by the ADC and stored in memory.

The Acquire Routine was loaded into memory and an analog signal was applied to the ADC using the VRS. The program was started, allowed to run for at least four cycles of the ADD and stopped manually.

The data stored in addresses 0130 and 0131 were recorded. The voltage was changed and the above process was repeated. The result of this data acquisition experiment is presented in Table 7 and a graph, denoting the linearity of the conversion, is depicted in Figure 16.

It was possible, therefore, to conclude that the voltages were directly related to their digitized values as produced by the ADC, and any change in voltage vs. a change in its corresponding digital value is linear.

Refer to Table 8. Notice that the first three columns contain the same data as used and/or calculated in the off-line simulation. It was hoped that, by using the same data, the direct relationship between the two methods would be shown.

In column one of Table 8, every tenth sample is a control sample, one of which was purposely put out of control in an effort to test all loops of the program. Column four contains voltages as calculated from the % T in column three, defining 100% T as a maximum 6 Volts; the other voltages follow naturally.

TABLE 7

RESULTS OF DATA ACQUISITION EXPERIMENT

VOLTAGE	CONVERSION GAIN	OCTAL HIGH	OCTAL LOW	DECIMAL CONVERSION
1.000	1024	0000	0163	0115
2.000	1024	0000	0346	0230
3.000	1024	0000	0527	0343
4.000	1024	0000	0704	0454
6.000	1024	0000	1251	0681
7.000	1024	0000	1443	0803
4.000	1024	0000	0705	0453
4.000	1024	0000	0704	0452
4.000	512	0000	0331	0217
6.000	1024	0000	1251	0681
7.000	2048	0000	3132	1626
7.000	4096	0000	6271	3257
8.000	4096	0000	7316	3790

The values in the OCTAL HIGH/LOW columns represent the values as received by the ND 812 minicomputer from the ADC, and stored in addresses represented by the symbolic addresses HIGH and LOW.

These octal numbers were then converted into their decimal equivalents, which are found in column 5.

The CONVERSION GAIN was changed for some of the values and a corresponding change in the value recorded reflects that change.

Some voltages were repeated to show constancy of the digitization process.

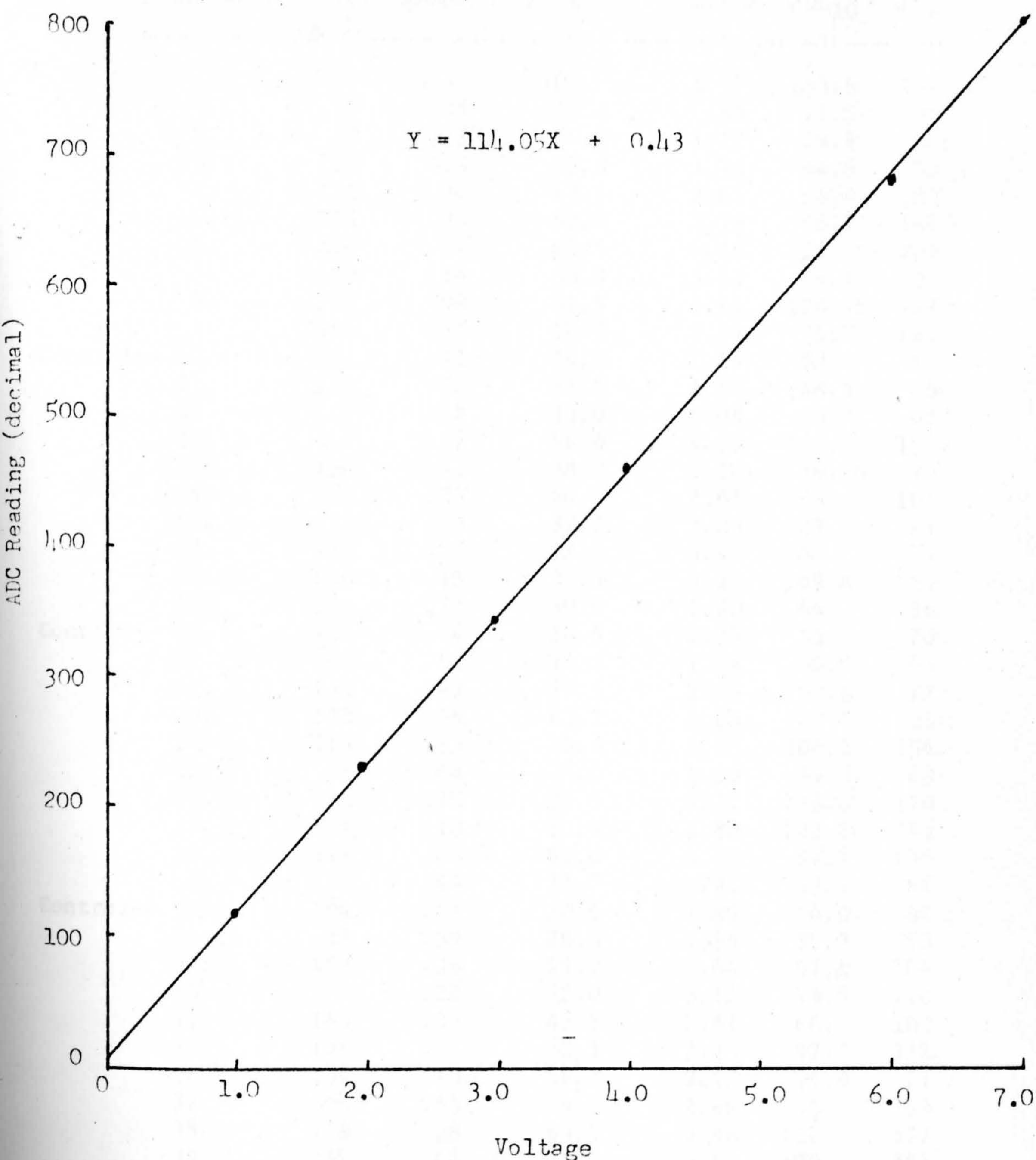


Figure 16 Linearity of Conversion

TABLE 8

DATA FOR SIMULATION 2

	STANDARD #	CONC. mg/dl	ABSORB.	% TRANS.	VOLTAGE	ADC_{10}	ADC_8
	---	---	0	100	6.0	153.6	231
	1.	0	.85	14.0	.84	21.5	26
	2.	50	.71	19.5	1.17	29.9	36
	3.	100	.54	29.0	1.74	44.5	55
	4.	150	.36	43.5	2.61	66.8	103
	5.	200	.20	63.0	3.78	96.7	141
	6.	250	.07	85.5	5.13	131	203
	7.	189	.24	58.0	3.48	89.1	131
	8.	238	.08	82.5	4.95	126.72	177
	9.	156	.31	49.0	2.94	75.2	115
Control--	10.	122	.47	34.0	2.04	52	64
	11.	110	.50	31.5	1.89	48.3	60
	12.	116	.48	33.0	1.98	50.7	62
	13.	210	.17	68.0	4.08	104.5	151
	14.	134	.42	38.0	2.28	58.36	70
	15.	156	.35	44.3	2.67	68	104
	16.	133	.43	37.2	2.23	27	71
	17.	189	.24	58.0	3.48	89	131
	18.	216	.15	71.5	4.29	109.8	156
	19.	104	.52	30.0	1.80	46	56
Control--	20.	128	.44	36.5	2.19	56	70
	21.	74	.62	24.0	1.44	36.8	45
	22.	132	.43	37.5	2.25	57.6	72
	23.	153	.36	43.7	2.62	67.1	103
	24.	216	.15	70.5	4.23	108.2	154
	25.	68	.64	23.0	1.38	35.3	43
	26.	232	.10	78.5	4.71	120.3	170
	27.	232	.10	80.0	4.80	122.8	173
	28.	188	.24	57.0	3.42	87.5	130
	29.	114	.49	32.0	1.92	49.1	61
Control--	30.	104	.52	30.0	1.80	46.0	56
	31.	83	.59	26.0	1.56	39.9	50
	32.	153	.36	44.0	2.64	67.6	104
	33.	177	.28	52.0	3.12	79.9	120
	34.	153	.36	43.5	2.61	66.8	103
	35.	196	.22	60.5	3.63	92.9	135
	36.	132	.43	37.0	2.22	56.8	71
	37.	95	.55	28.0	1.68	43	53
	38.	238	.08	83.0	4.98	127	177
	39.	238	.08	84.0	5.04	129	201
Control--	40.	120	.47	34.0	2.04	52	64

Column five represents decimal equivalents of the digitized, octal number as would be produced by the ADC. It was found to be much easier to calculate the decimal values for each % T and then convert these to their octal equivalents (found in column six) for the purposes of this simulation. A minor problem of rounding off numbers cropped up here, as will be illustrated below.

A data tape, containing the octal values listed in column six, was prepared on the E & L Micro-Designer currently housed in room 329 of the Ward-Beecher Science Building at Youngstown State University. This same tape may also be prepared by the ND 812 by using the Octal Debug program.

After this data tape was read into memory Field 2, the Octal Debug program was used to print the values to verify the compatability of the E & L Micro-Designer with the ND 812. This printout appears in Listing 18. The first column represents the initial address for the row. The data in the second column will be found at this initial address; the remaining seven columns will be found in the next seven sequential addresses. In other words, the data are addressed sequentially across the rows, with the first column representing the starting address of that row. (Addresses are listed in octal numbers as are the data.) By comparing Listing 18 with the sixth row of Figure 21, one can see that the data tape was indeed compatible.

These data were then used in a dry run simulation of the Auto-Analyzer Data Handler Program. The output of this run is given in Listing 19. Notice that the values for Glucose concentration correspond closely with those in Listing 21, but are not exactly equivalent. This is due to the round-off problem previously mentioned.

Also, note that every tenth sample, (control) is tested to check for system malfunction. If the calculated value drifts too far, the computer will issue an "OUT OF CONTROL" warning, stop execution, and wait for instructions by the operator to continue (operator types a one, (1)) or abort (operator types a zero, (0)) execution.

The next step was to test the assembly-language AutoAnalyzer Routine. It was pretested once, working successfully, and thereafter failing to operate. It is believed that this is due to a known computer hardware malfunction, quite possibly one of the IC's controlling functions involving the K register. Other problems such as the inability to load the magnetic tape programs also occurred during this time period.

This program, when the computer was functioning properly, would store data for 100% T in Field 2, location 0000; data for the baseline in Field 2, location 0001, and data for each of the next forty standards and samples in the next forty successive locations. At this point execution would stop, the AutoAnalyzer Data Handler program would be read into memory Field 0, and data processing would begin.

The Octal Debug program may be used again here to make a hard-copy record of the data, should they inadvertantly be erased before they are no longer needed.

One final attempt was made to try an actual run of the AutoAnalyzer with the AutoAnalyzer Acquire Routine. Once again, the computer failed to function, which was proven by attempting to execute programs which were known to operate, and the test was aborted.

CHAPTER VII

SUMMARY

Development of computer aided experimentation is, by no means, a new field in chemistry, but rather an ever-widening area of interest. With more chemists becoming interested in computer technology, computers have made their way into the deepest regions of chemical research. Analytical chemists, among others, have recognized the value of having a digital computer in the laboratory.

Any device, no matter what physical size or design, which outputs an analog signal, can be interfaced to a computer. In addition to the AutoAnalyzer, this technique can be applied to gas chromatography, flame spectroscopy, fluorometry, atomic absorption, mass spectroscopy, and the like, with comparable success and reliability. This research problem was offered as an example of the myriad possible applications.

It is the first and foremost recommendation of this writer that all future computer applications be totally independent from the computer center, if at all possible. This research was greatly hindered and often obstructed by the computer center.

When computer malfunctions were reported to the computer center, these reports were ignored. When the computer finally broke down completely, more than four months elapsed before the computer center even gave any consideration to the problem. When the problem with the computer was diagnosed correctly by this researcher, the computer center chose to ignore this diagnosis which eventually resulted in a substantial financial loss. The repairman's diagnosis confirmed that of this researcher and additional time and money was lost while waiting for replacement parts he could have brought with him, if he had known the problem in advance.

This incident should illustrate further the need for the programmer to have complete comprehension of the diagnostic programs available for his computer.

The failure of the AutoAnalyzer Acquire program to operate was caused by a known computer hardware problem and not due to logic errors in the program itself.

Finally, future development of this problem is totally open-ended. Total computer control of the AutoAnalyzer is possible. That is, have the computer monitor the data and make adjustments automatically, as needed. These adjustments include: flow rates, pump speeds, bath temperatures and sampling rates. The computer could also flag out of range data, thus alerting the technician to a possible diabetic or hypoglycemic patient. In the event that the control values have drifted, the computer could: stop operation, suggest a possible diagnosis and repair procedure to the technician, or, if the drift can be corrected by a minor adjustment as mentioned above, make such an adjustment.

Future plans also include using the E & L Micro-Designer as the primary data acquisition device and function monitor, thus releasing the ND 812 from all but data processing responsibility.

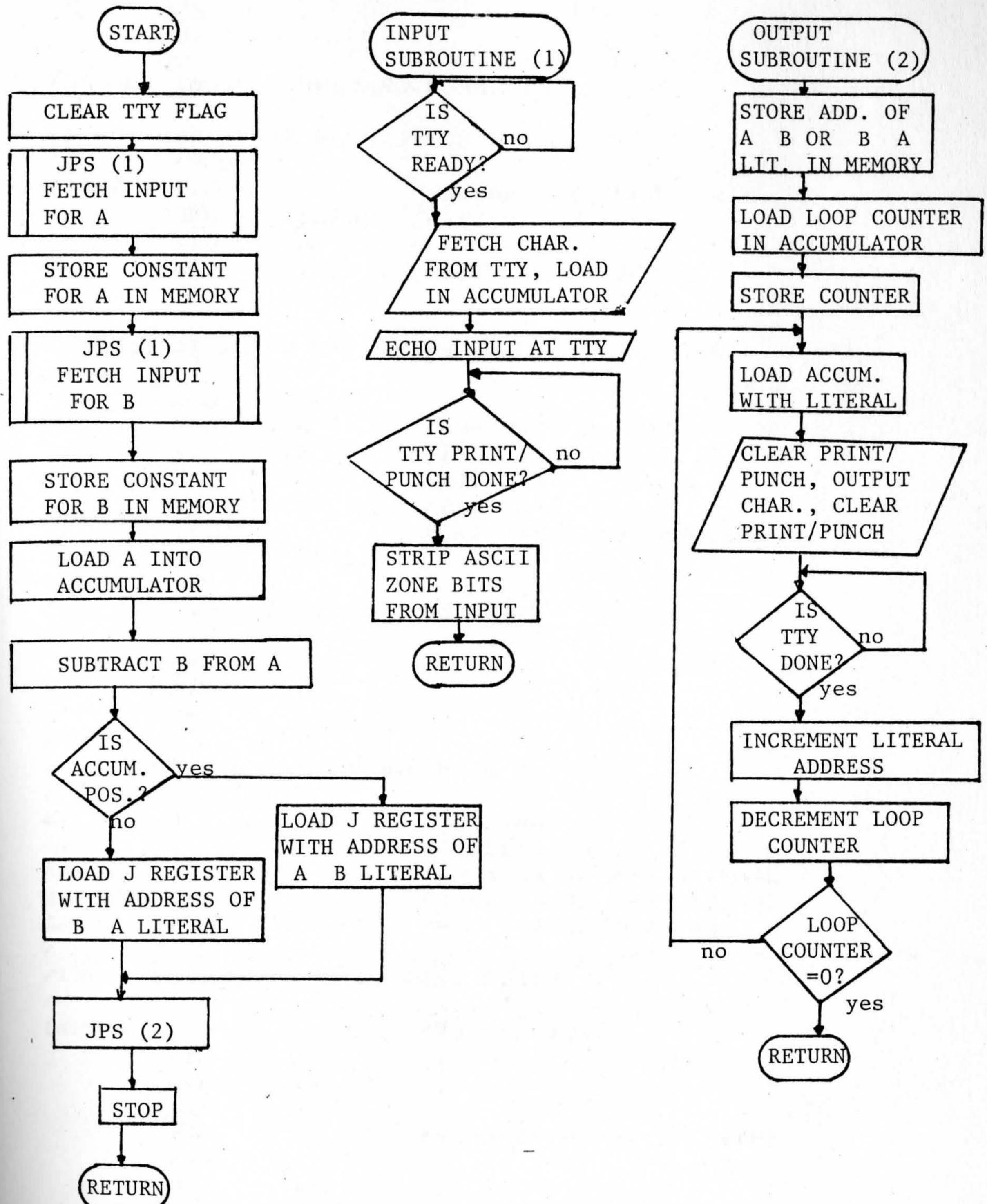
Ultimately, the total operation is to be done by the E & L Micro-Designer which, by the way, is solely owned, operated and maintained by the Chemistry Department, as all laboratory devices should be.

Much work remains before these implementations will become realities. Through this work, the possible number of spin-off applications are, as mentioned above, myriad.

APPENDIX A

Listings of Flowcharts and Programs

LISTING PAGE



Listing 1. Flowchart A > B[57]

```
/LABEL   INSTR   OPERAND COMMENTS
/
/INPUT AND STORE VALUES FOR A & B
         *200
START,   TIF              /CLEAR TTY FLAG
         JPS     INPUT    /GET VALUE FOR A
         STJ     A
         JPS     INPUT    /GET VALUE FOR B
         STJ     B
/
/DETERMINE WHICH OF THE TWO VALUES IS LARGER
/
         LDJ     A
         SBJ     B        /SUBTRACT B FROM A
         SIP     J        /TEST FOR A POSITIVE
         JMP     BRAN     /NO! B>A
         LDJ     ABCST    /YES! A>B
         SKIP             /SKIP NEXT INSTRUCTION
BRAN,    LDJ     BACST
/
/SET UP AND OUTPUT EXPRESSION
/
         JPS     OUT
         STOP
         JMP     START
/
/WORKING OR DATA STORAGE AREA
/
A,       0                /CONSTANT A
B,       0                /CONSTANT B
ABCST,   AB               /ADDRESS OF A>B LITERAL
BACST,   BA               /ADDRESS OF B>A LITERAL
C260,    260              /ASCII ZONE CONSTANT
/
/INPUT ROUTINE + ASCII ZONE STRIP
/
INPUT,   0                /ENTRY POINT
         TIS
         JMP     .-1
         TRF
         TCP              /ECHO INPUT AT TELETYPE
         TOS
         JMP     .-1
         SBJ     C260
         JMPG    INPUT
/
/OUTPUT ROUTINE - OUTPUT ASCII EXPRESSION
/
OUT,     0                /ENTRY POINT
```

Listing 2. Assembly Language Program A > B[58]

```
        STJ     LOOP+1
        LDJ     C5          /SET NUMBER OF CHARACTER
        STJ     CTR
/
/OUTPUT DATA LOOP
/
LOOP,   TWLDJ
        0
        TCP
        TOS
        JMP     .-1
        ISZ     LOOP+1
        DSZ     CTR         /TEST FOR ALL CHARACTERS OUT
        JMP     LOOP        /NO
        JMP@    OUT         /RETURN
C5,     5
CTR,    0
/
/OUTPUT MESSAGES
/
AB,     215
        212
        301                 /A
        276                 />
        302                 /B
BA,     215
        212
        302                 /B
        276                 />
        301                 /A
$                           /END CHARACTER
```

Listing 2. Assembly Language Program A>B (cont.)

```
/LABEL INSTR.  OPERAND COMMENTS
/
/INPUT AND STORE VALUES FOR A & B
                        *200
  0200   7401   START,  TIF                /CLEAR TTY FLAG
  0201   6423           JPS      INPUT     /GET VALUE FOR A
  0202   5415           STJ      A
  0203   6421           JPS      INPUT     /GET VALUE FOR B
  0204   5414           STJ      B
/
/DETERMINE WHICH OF THE TWO VALUES IS LARGER
/
  0205   5012           LDJ      A
  0206   4012           SBJ      B         /SUBTRACT B FROM A
  0207   1502           SIP      J         /TEST FOR A POSITIVE
  0210   6003           JMP      BRAN      /NO! B>A
  0211   5010           LDJ      ABCST     /YES! A>B
  0212   1442           SKIP               /SKIP NEXT INSTRUCTION
  0213   5007   BRAN,   LDJ      BACST
/
/SET UP AND OUTPUT EXPRESSION
/
  0214   6421           JPS      OUT
  0215   0000           STOP
  0216   6116           JMP      START
/
/WORKING OR DATA STORAGE AREA
/
  0217   0000   A,      0                  /CONSTANT A
  0220   0000   B,      0                  /CONSTANT B
  0221   0254   ABCST,  AB                 /ADDRESS OF A>B LITERAL
  0222   0261   BACST,  BA                 /ADDRESS OF B>A LITERAL
  0223   0260   C260,   260                /ASCII ZONE CONSTANT
/
/INPUT ROUTINE + ASCII ZONE STRIP
/
  0224   0000   INPUT,  0                  /ENTRY POINT
  0225   7404           TIS
  0226   6101           JMP      .-1
  0227   7403           TRF
  0230   7413           TCP                /ECHO INPUT AT TELETYPE
  0231   7414           TOS
  0232   6101           JMP      .-1
  0233   4110           SBJ      C260
  0234   6310           JMP@     INPUT
/
/OUTPUT ROUTINE - OUTPUT ASCII EXPRESSION
/
  0235   0000   OUT,    0                  /ENTRY POINT
```

Listing 3. Assembled Program A > B[59]

```
        0236   5404            STJ      LOOP+1
        0237   5013            LDJ      C5        /SET NUMBER OF CHARACTER
        0240   5413            STJ      CTR
/
/OUTPUT DATA LOOP
/
        0241   0500   LOOP,    TWLDJ
        0242   0000            0
        0243   7413            TCP
        0244   7414            TOS
        0245   6101            JMP      .-1
        0246   3504            ISZ      LOOP+1
        0247   3004            DSZ      CTR       /TEST FOR ALL CHARACTERS OUT
        0250   6107            JMP      LOOP      /NO
        0251   6314            JMP@     OUT       /RETURN
        0252   0005   C5,      5
        0253   0000   CTR,     0
/
/OUTPUT MESSAGES
/
        0254   0215   AB,      215
        0255   0212            212
        0256   0301            301                /A
        0257   0276            276                />
        0260   0302            302                /B
        0261   0215   BA,      215
        0262   0212            212
        0263   0302            302                /B
        0264   0276            276                />
        0265   0301            301                /A
```

Listing 3. Assembled Program A > B (cont.)

```
SE 1200
A       = 0217
AB      = 0254
ABCST   = 0221
B       = 0220
B4      = 0261
BACST   = 0222
BRAN    = 0213
C260    = 0223
C5      = 0252
CTR     = 0253
INPUT   = 0224
LOOP    = 0241
OUT     = 0235
START   = 0200
ER 0000
```

Listing 3. Assembled Program A > B (cont.)

```
12
B>A42
A>B84
A>B50
A>B36
B>A42
A>B
```

Listing 4. Program A>B Run

```
                        *200
0200  7411  START,      TOC
0201  5001              LDJ      COLHD
0202  0206              COLHD,   AB
0203  6456              JPS      OUT
0204  0000              STOP
0205  6105              JMP      START
/
/COLUMN HEADINGS
/
0206  0215  AB,         215                /CR
0207  0212              212                /LF
0210  0212              212                /LF
0211  0212              212                /LF
0212  0322              322                /R
0213  0305              305                /E
0214  0324              324                /T
0215  0256              256                /.
0216  0240              240                /SP
0217  0324              324                /T
0220  0311              311                /I
0221  0315              315                /M
0222  0305              305                /E
0223  0240              240                /SP
0224  0250              250                /(
0225  0323              323                /S
0226  0305              305                /E
0227  0303              303                /C
0230  0323              323                /S
0231  0251              251                /)
0232  0240              240                /SP
0233  0320              320                /P
0234  0313              313                /K
0235  0256              256                /.
0236  0240              240                /SP
0237  0310              310                /H
0240  0324              324                /T
0241  0256              256                /.
0242  0240              240                /SP
0243  0240              240                /SP
0244  0320              320                /P
0245  0313              313                /K
0246  0256              256                /.
0247  0240              240                /SP
0250  0301              301                /A
0251  0322              322                /R
0252  0305              305                /E
0253  0301              301                /A
0254  0215              215                /CR
0255  0212              212                /LF
```

Listing 5. Program "Output"

```
     0256    0212              212                       /LF
     0257    0212              212                       /LF
/
     0260    0305              305                       /E
/
/OUTPUT ROUTINE
/
     0261    0000    OUT,      0                         /ENTRY POINT
     0262    5404              STJ        LOOP+1
     0263    5013              LDJ        C5
     0264    5413              STJ        CTR
/
/OUTPUT DATA LOOP
/
     0265    0500    LOOP,     TWLDJ
     0266    0000              0
     0267    7413              TCP
     0270    7414              TOS
     0271    6101              JMP        .-1
     0272    3504              ISZ        LOOP+1
     0273    3004              DSZ        CTR
     0274    6107              JMP        LOOP
     0275    6314              JMP@       OUT
     0276    0054    C5,       54
     0277    0000    CTR,      0
```

Listing 5. Program "Output" (cont.)

```
RET. TIME (SECS) PK. HT.  PK. AREA

E
```

Listing 6. Program "Output" Run

```
                                      *100
0100   0000   START,   0
0101   7401            TIF
0102   6422            JPS        INPUT
0103   5415            STJ        X
0104   6420            JPS        INPUT
0105   5414            STJ        Y
0106   5012            LDJ        X
0107   1204            LKFJ
0110   5011            LDJ        Y
0111   1000            MPY
0112   1302            LJKFRS
0113   1120            AJK J
0114   5406            STJ        Z
0115   6420            JPS        OUT
0116   0000            STOP
0117   6517            JPS        START
0120   0000   X,       0
0121   0000   Y,       0
0122   0000   Z,       0
0123   0260   C260,    260
/
0124   0000   INPUT,   0
0125   7404            TIS
0126   6101            JMP        .-1
0127   7403            TRF
0130   7413            TCP
0131   7414            TOS
0132   6101            JMP        .-1
0133   4110            SBJ        C260
0134   6310            JMP0       INPUT
/
0135   0000   OUT,     0
0136   5036            LDJ        CR
0137   6442            JPS        OP
0140   5035            LDJ        LF
0141   6440            JPS        OP
0142   5122            LDJ        X
0143   4520            ADJ        C260
0144   6435            JPS        OP
0145   5031            LDJ        SP
0146   6433            JPS        OP
0147   5030            LDJ        M
0150   6431            JPS        OP
0151   5025            LDJ        SP
0152   6427            JPS        OP
0153   5132            LDJ        Y
0154   4531            ADJ        C260
0155   6424            JPS        OP
0156   5020            LDJ        SP
```

Listing 7. Program "Multiply"

```
0157   6422              JPS     OP
0160   5020              LDJ     E
0161   6420              JPS     OP
0162   5014              LDJ     SP
0163   6416              JPS     OP
0164   5142              LDJ     Z
0165   4542              ADJ     C260
0166   6413              JPS     OP
0167   5005              LDJ     CR
0170   6411              JPS     OP
0171   5004              LDJ     LF
0172   6407              JPS     OP
0173   6336              JMP@    OUT
0174   0215   CR,        215
0175   0212   LF,        212
0176   0240   SP,        240
0177   0330   M,         330
0200   0275   E,         275
/
0201   0000   OP,        0
0202   7413              TCP
0203   7414              TOS
0204   6101              JMP     .-1
0205   6304              JMP@    OP


SE 1174
C260    = 0123
CR      = 0174
E       = 0200
INPUT   = 0124
LF      = 0175
M       = 0177
OP      = 0201
OUT     = 0135
SP      = 0176
START   = 0100
X       = 0120
Y       = 0121
Z       = 0122
ER 0000
```

Listing 7. Program "Multiply" (cont.)

```
10
1 X 0 = 0
11
1 X 1 = 1
12
1 X 2 = 2
13
1 X 3 = 3
14
1 X 4 = 4
15
1 X 5 = 5
16
1 X 6 = 6
17
1 X 7 = 7
18
1 X 8 = 8
19
1 X 9 = 9
20
2 X 0 = 0
21
2 X 1 = 2
22
2 X 2 = 4
23
2 X 3 = 6
24
2 X 4 = 8
20
2 X 0 = 0
30
3 X 0 = 0
31
3 X 1 = 3
32
3 X 2 = 6
33
3 X 3 = 9
```

Listing 8. Program "Multiply" Run

```
/                                /ACQUIRE ROUTINE 1
/
                         RDIB =   7722
                         *1
0001  0000  START,       0
0002  5436               STJ      SJ       /GENERAL INTERRUPT HANDLER
0003  0550               TWSTK             /SAVE REGISTERS
0004  0041               SK
0005  1302               LJKFRS
0006  5434               STJ      SR
0007  0550               TWSTK
0010  0043               SS
0011  1011               LJST
0012  5432               STJ      STATUS
/
0013  0740               TWIO
0014  0606               0606              /READ 4410 STATUS
0015  1204               LKFJ
0016  2027               ANDF     P3400
0017  1501               SNZ      J        /LOOK AT INTERRUPTS
0020  0000               STOP              /4400 INTERRUPT
0021  1142               SFTZ     J    2   /BAD INTERRUPT
0022  1502               SIP      J
0023  6023               JMP      CLOCK    /4400 CLOCK
/
0024  5020  DONE,        LDJ      STATUS   /RELOAD REGISTERS
0025  1002               RFOV
0026  7722               RDIB
0027  5013               LDJ      SR
0030  0510               TWLDK
0031  0043               SS
0032  1301               LRSFJK
0033  5005               LDJ      SJ
0034  0510               TWLDK
0035  0041               SK
0036  1004               IONH
0037  6336               JMPD     START    /TURN HIGH LEVEL ON
0040  0000  SJ,          0
0041  0000  SK,          0
0042  0000  SR,          0
0043  0000  SS,          0
0044  0000  STATUS,      0
0045  3400  P3400,       3400
/
0046  0740  CLOCK,       TWIO
0047  0612               0612              /CLEAR CLOCK
0050  6124               JMP      DONE

                         *60
```

Listing 9. Program "Acquire"

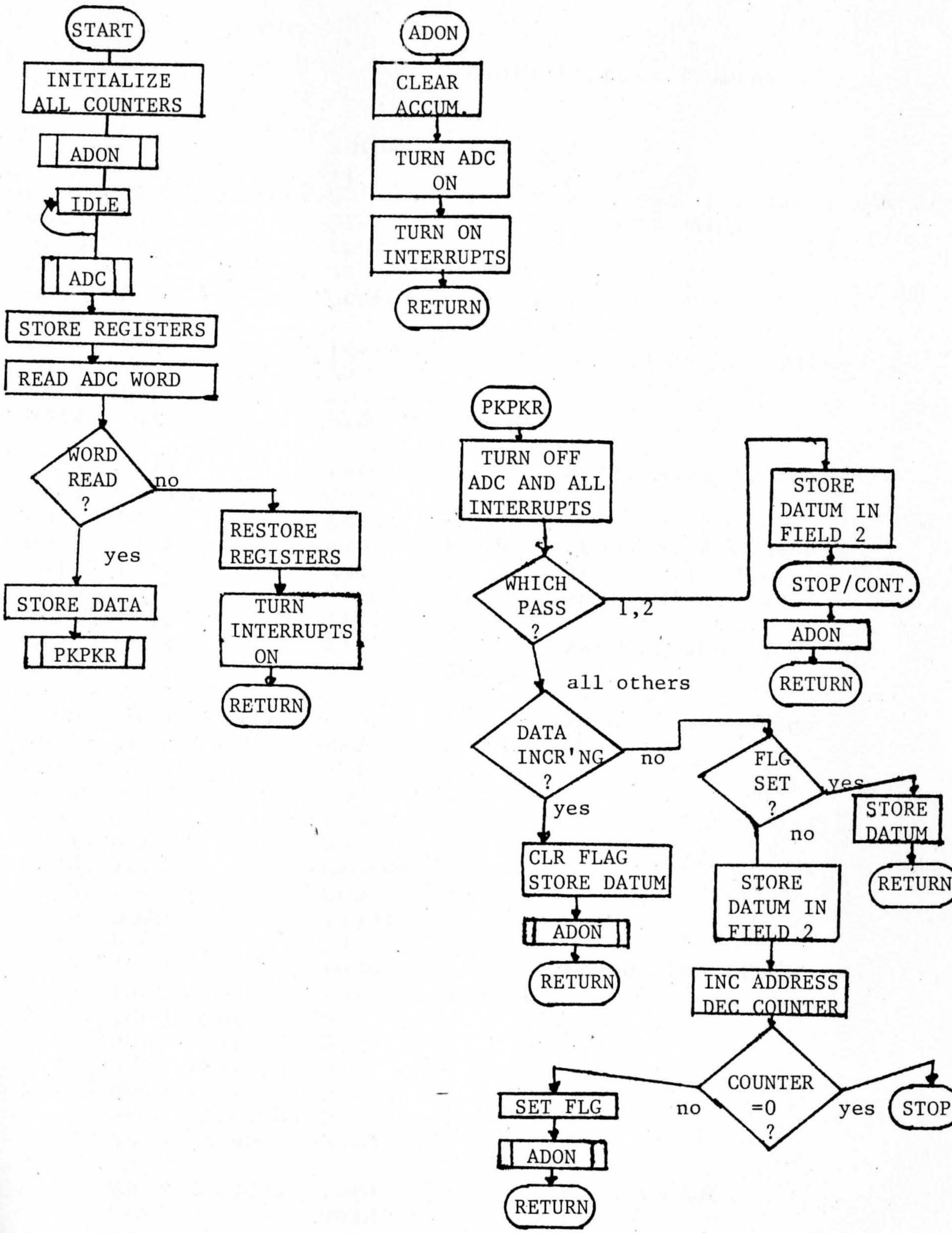
```
0060  0000  ADON,   0
0061  1510          CLR     J
0062  0510          TWLDK
0063  0070          ADCMD
0064  0740          TWIO
0065  2026          2026
0066  1004          IONA
0067  6307          JMP@    ADON
0070  0020  ADCMD,  0020            /ADC ACCUM. MODE
/
0071  0000  ADOFF,  0               /ADC OFF ROUTINE
0072  1003          IOFF
0073  1510          CLR     J
0074  1610          CLR     K
0075  0740          TWIO
0076  2026          2026
0077  6306          JMP@    ADOFF
/
                    *101
0101  0000  ADC,    0               /ADC INTERRUPT PROCESS
0102  5423          STJ     ADCJ    /STORE REGISTERS
0103  0550          TWSTK
0104  0126          ADCK
0105  1011          LJST
0106  5421          STJ     ADCST
0107  1450          CLR     O
0110  7527          7527            /READ ADC WORD
0111  1602          STP     K
0112  6004          JMP     ADC1
0113  0550          TWSTK
0114  0130          HIGH
0115  5414          STJ     LOW
0116  0510  ADC1,   TWLDK           /RESTORE REGISTERS
0117  0126          ADCK
0120  5007          LDJ     ADCST
0121  1002          RFOV
0122  5003          LDJ     ADCJ
0123  1004          IONA            /TURN INTERRUPT ON
0124  6323          JMP@    ADC
0125  0000  ADCJ,   0
0126  0000  ADCK,   0
0127  0000  ADCST,  0
0130  0000  HIGH,   0
0131  0000  LOW,    0
                    *200
0200  0640  BEGIN,  TWJPS
0201  0060          ADON
0202  1400          IDLE
0203  6101          JMP     .-1
0204  0640          TWJPS
```

Listing 9. Program "Acquire" (cont.)

```
0205   0071          ADOFF
0206   0000          STOP
0207   6107          JMP       BEGIN
```

```
SE 1234
ADC        = 0101
ADC1       = 0116
ADCJ       = 0125
ADCK       = 0126
ADCXD      = 0070
ADCST      = 0127
ADOFF      = 0071
ADON       = 0050
BEGIN      = 0200
CLOCK      = 0046
DONE       = 0024
HIGH       = 0130
LOW        = 0131
P3400      = 0045
RJIB       = 7722
SJ         = 0040
SK         = 0041
SR         = 0042
SS         = 0043
START      = 0001
STATUS     = 0044
ER 0000
```

Listing 9. Program "Acquire" (cont.)



Listing 10. Flowchart for Program "AutoAnalyzer Routine"

```
/
/                                          /AUTOANALYZER ROUTINE

                              RJIB=    7722
                              *1
0001  0000  HANDLR,  0                     /GENERAL INTERRUPT HANDLER
0002  5436           STJ       SJ          /SAVE REGISTERS
0003  0550           TWSTK
0004  0041           SK
0005  1302           LJKFRS
0006  5434           STJ       SR
0007  0550           TWSTK
0010  0043           SS
0011  1011           LJST
0012  5432           STJ       STATUS
/
0013  0740           TWIO                  /READ 4410 STATUS
0014  0606           0606
0015  1204           LKFJ
0016  2027           ANDF      P3400       /LOOK AT INTERRUPTS
0017  1501           SNZ       J1          /4400 INTERRUPT
0020  0000           STOP                  /BAD INTERRUPT
0021  1142           SFTZ      J  2
0022  1502           SIP       J           /4400 CLOCK
0023  6023           JMP       CLOCK
/
0024  5030  DONE,    LDJ       STATUS      /RELOAD REGISTERS
0025  1002           RFOV
0026  7722           RJIB
0027  5013           LDJ       SR
0030  0510           TWLDK
0031  0043           SS
0032  1301           LRSFJK
0033  5005           LDJ       SJ
0034  0510           TWLDK
0035  0041           SK
0036  1004           IONH                  /TURN HIGH LEVEL ON
0037  6336           JMPJ      HANDLR
0040  0000  SJ,      0
0041  0000  SK,      0
0042  0000  SR,      0
0043  0000  SS,      0
0044  0000  STATUS,  0
0045  3400  P3400,   3400
/
0046  0740  CLOCK,   TWIO                  /CLEAR CLOCK
0047  0612           0612
0050  6124           JMP       DONE
/
                     *60
```

Listing 11. Program "AutoAnalyzer Routine"

```
0060  0000  ADON,   0
0061  1510          CLR     J
0062  0510          TWLDK
0063  0070          ADCMD
0064  0740          TWIO
0065  2026          2026
0066  1004          IONH
0067  6307          JMP@    ADON
0070  0020  ADCMD,  0020            /ADC ACCUM. MODE
/
                    *101
0101  0000  ADC,    0
0102  5425          STJ     ADCJ    /ADC INTERRUPT PROCESS
0103  0550          TWSTK
0104  0130          ADCK
0105  1011          LJST
0106  5423          STJ     ADCST
0107  1450          CLR     0
0110  7527          7527            /READ ADC WORD
0111  1602          SIP     K
0112  6006          JMP     ADC1
0113  0550          TWSTK
0114  0132          HIGH
0115  5416          STJ     LOW
0116  0640          TWJPS
0117  0222          PKPAR
0120  0510  ADC1,   TWLDK           /RESTORE REGISTERS
0121  0130          ADCK
0122  5007          LDJ     ADCST
0123  1002          RFOV
0124  5003          LDJ     ADCJ
0125  1004          IONH            /TURN INTERRUPT ON
0126  6325          JMP@    ADC
0127  0000  ADCJ,   0
0130  0000  ADCK,   0
0131  0000  ADCST,  0
0132  0000  HIGH,   0
0133  0000  LOW,    0
/
/
                    *0200
0200  5020  BEGIN,  LDJ     PKNO    /INITIALIZE ALL COUNTERS
0201  0540          TWSTJ
0202  0325          PKCTR
0203  5013          LDJ     C1
0204  5413          STJ     CTR
0205  0540          TWSTJ
0206  0254          STCTR
0207  5012          LDJ     ADRES
0210  0540          TWSTJ
```

Listing 11. Program "AutoAnalyzer Routine" (cont.)

```
0211  0271          ADDRS
0212  0640          TWJPS
0213  0060          ADON               /TURN ADC ON
0214  1400          IDLE               /WAIT FOR INTERRUPT
0215  6101          JMP     .-1
0216  0001  C1,     1
0217  0000  CTR,    0
0220  0050  PKNO,   50
0221  0000  ADRES,  0
/
/
0222  0000  PKPKR,  0
0223  1003          IOFF               /ADC OFF ROUTINE
0224  1510          CLR     J
0225  1610          CLR     K
0226  0740          TWIO
0227  2026          2026
0230  3111          DSZ     CTR
0231  6002          JMP     DATA       /DATA PTS.
0232  6040          JMP     STNDRD     /100% T
0233  3031  DATA,   DSZ     STCTR
0234  6002          JMP     PKS
0235  6035          JMP     STNDRD     /BASELINE
0236  0500  PKS,    TWLDJ
0237  0133          LOW
0240  5430          STJ     TEMP2
0241  0400          TWSBJ              /LOW-TEMP1
0242  0267          TEMP1
0243  1506          SIN     J
0244  6011          JMP     STORE      /PK RISING
0245  5021          LDJ     FLG        /PK FALLING
0246  1501          SNZ     J          /FLAG SET?
0247  6035          JMP     STPK       /NO, NEW PEAK
0250  5020          LDJ     TEMP2
0251  5416          STJ     TEMP1
0252  0640          TWJPS
0253  0060          ADON
0254  6332          JMP@    PKPKR
0255  5010  STORE,  LDJ     ZERO       /CLEAR FLAG
0256  5410          STJ     FLG
0257  5011          LDJ     TEMP2
0260  5407          STJ     TEMP1
0261  0640          TWJPS
0262  0060          ADON
0263  6341          JMP@    PKPKR
0264  0000  STCTR,  0
0265  0000  ZERO,   0
0266  0000  FLG,    0
0267  0000  TEMP1,  0
0270  0000  TEMP2,  0
```

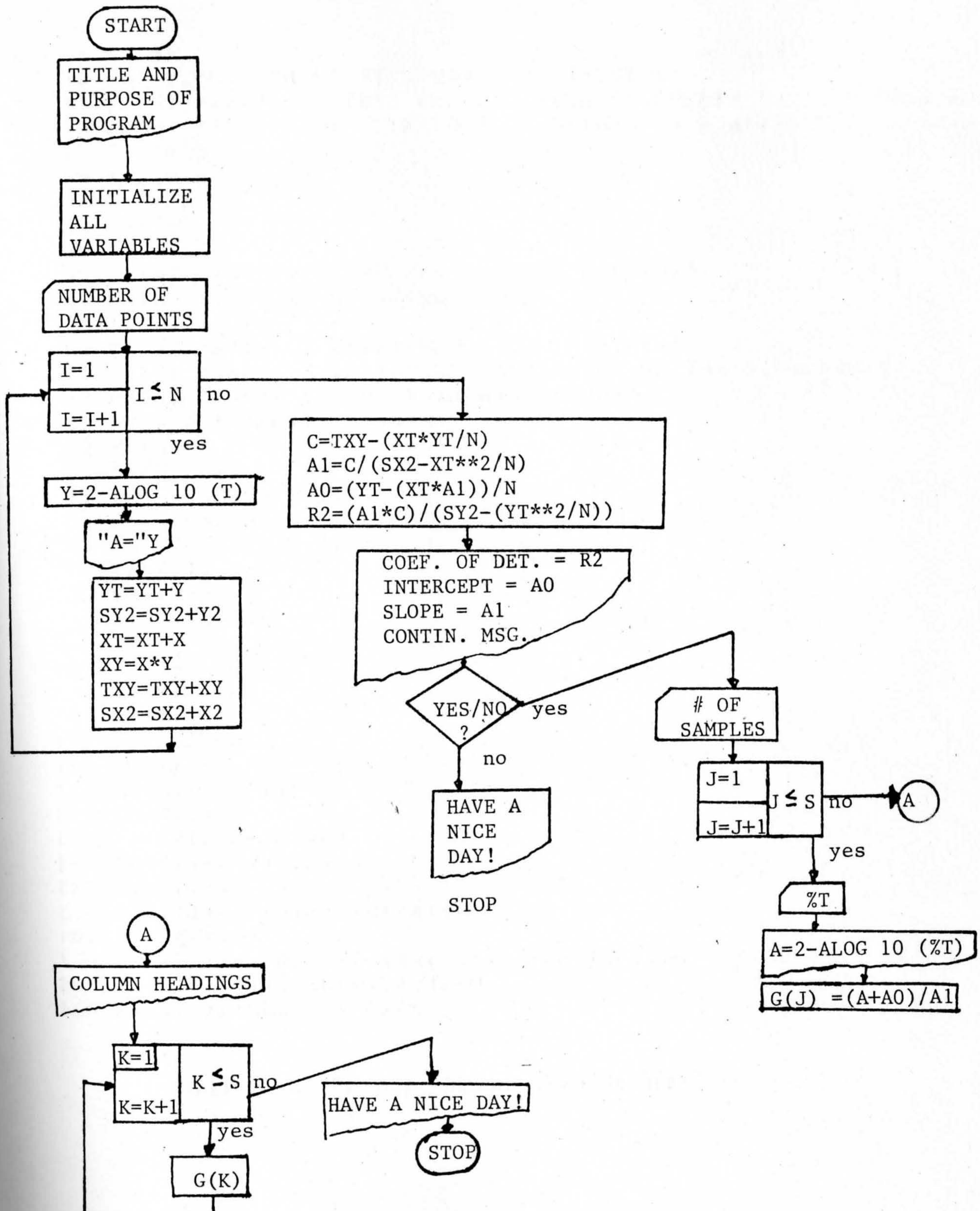
Listing 11. Program "AutoAnalyzer Routine" (cont.)

```
0271   0000   ADDRS,   0
/
/
0272   0500   STNDRD,  TWLDJ
0273   0133            LOW                 /BASELINE OR 100% T
0274   0566            0566                /TWSTJ@ FIELD 02
0275   0271            ADDRS
0276   3505            ISZ       ADDRS
0277   1400            IDLE
0300   0000            STOP                /OPERATOR OK
0301   0640            TWJPS
0302   0060            ADON
0303   6361            JMP@      PKPKR
/
/
0304   0500   STPK,    TWLDJ
0305   0267            TEMP1
0306   0566            0566                /TWSTJ@ FIELD 02
0307   0271            ADDRS
0310   3517            ISZ       ADDRS
0311   1400            IDLE
0312   3013            DSZ       PKCTR
0313   6002            JMP       CONT
0314   6012            JMP       FINIS
0315   5007   CONT,    LDJ       ONE
0316   0540            TWSTJ               /SET FLAG
0317   0266            FLG
0320   0640            TWJPS
0321   0060            ADON
0322   0620            TWJMP@
0323   0222            PKPKR
0324   0001   ONE,     0001
0325   0000   PKCTR,   0
/
/
0326   0000   FINIS,   STOP
0327   0600            TWJMP
0330   0200            BEGIN
```

Listing 11. Program "AutoAnalyzer Routine" (cont.)

```
SE 1350
ADC     = 0101
ADC1    = 0120
ADCJ    = 0127
ADCK    = 0130
ADCMD   = 0070
ADCST   = 0131
ADDRS   = 0271
ADON    = 0060
ADRES   = 0221
BEGIN   = 0200
C1      = 0216
CLOCK   = 0046
CONT    = 0315
CTR     = 0217
DATA    = 0233
DONE    = 0024
FINIS   = 0326
FLG     = 0266
HANDLR  = 0001
HIGH    = 0132
LOW     = 0133
ONE     = 0324
P3400   = 0045
PKCTR   = 0325
PKNO    = 0220
PKPKR   = 0222
PKS     = 0236
RJIB    = 7722
SJ      = 0040
SK      = 0041
SR      = 0042
SS      = 0043
STATUS  = 0044
STCTR   = 0264
STNDRD  = 0272
STORE   = 0255
STPK    = 0304
TEMP1   = 0267
TEMP2   = 0270
ZERO    = 0265
ER 0000
```

Listing 11. Program "AutoAnalyzer Routine" (cont.)



Listing 12. Flowchart for "Linear Least Squares" Program

```
  5 ERASE
 10 PRINT 'LINEAR LEAST SQUARES REGRESSION'
 11 PRINT /,/,'        THIS PROGRAM WILL CALCULATE THE EQUATION FOR'
 12 PRINT 'THE "BEST" STRAIGHT LINE THROUGH A GIVEN SET OF DATA POIN
 15 SX2 = 0
 20 SY2 = 0
 25 XT = 0
 30 TXY = 0
 35 YT = 0
 40 PRINT /,/,'INPUT NUMBER OF DATA POINTS,'
 41 PRINT 'FOR THE STANDARD CURVE.'
 45 INPUT N
 50 PRINT /,'INPUT (X,Y) FOR EACH DATA PT.'
 51 PRINT 'WHERE X IS THE CONCENTRATION OF THE STANDARD,'
 52 PRINT 'AND Y IS THE % TRANSMISSION.'
 55 DO 110 I=1,N,1
 59 PRINT /
 60 INPUT X,T
 61 Y = 2-ALOG 10 (T)
 62 FMT (F5,3)
 63 PRINT '    A = ',Y
 65 YT=YT+Y
 70 Y2=Y**2
 75 SY2=SY2+Y2
 80 XT=XT+X
 85 XY=X*Y
 90 TXY=TXY+XY
 95 X2=X**2
100 SX2=SX2+X2
110 CONTINUE
115 C=TXY-(XT*YT/N)
120 S2X=XT**2
125 A1=C/(SX2-S2X/N)
130 A0=(YT-(XT*A1))/N
135 T2Y=YT**2
140 R2=(A1*C)/(SY2-(T2Y/N))
145 FMT (F6,3)
150 PRINT /,/,'COEFFICIENT OF DETERMINATION = ',R2
155 PRINT /,'INTERCEPT = ',A0
160 PRINT /,'SLOPE = ',A1
```

Listing 13. Program "Linear Least Squares"

```
165 PRINT /,/,/,'DO YOU WISH TO CALCULATE THE CONCENTRATIONS OF'
166 PRINT 'GLUCOSE FOR EACH SERUM SAMPLE?'
167 PRINT /,'TYPE "1" FOR YES, OR ZERO, "0", FOR NO.'
180 INPUT M
185 IF(M) 190,270,200
190 PRINT 'TRY AGAIN, PLEASE.'
195 GOTO 167
200 PRINT /,/,'INPUT NUMBER OF SAMPLES TO BE TESTED.'
205 INPUT S
209 FMT (I2)
210 PRINT /,'INPUT %TRANSMISSION FOR EACH OF THE ',S,' SAMPLES,'
211 PRINT 'ONE AT A TIME.'
214 FMT (F14,6)
215 DO 235 J=1,S,1
219 FMT (I2)
220 PRINT /,'SAMPLE # ',J
225 INPUT P
226 FMT (F5,3)
227 A = 2-ALOG 10 (P)
229 PRINT '    A = ',A
230 G(J) = (A-A0)/A1
235 CONTINUE
240 PRINT /,/,/,'    SAMPLE         GLUCOSE'
245 PRINT '    NUMBER    CONCENTRATION'
246 FMT (F9,2)
250 DO 260 K=1,S,1
255 PRINT K,'      ',G(K)
260 CONTINUE
270 PRINT /,/,/,'HAVE A NICE DAY!'
280 STOP
>
```

Listing 13. Program "Linear Least Squares" (cont.)

```
LINEAR LEAST SQUARES REGRESSION

     THIS PROGRAM WILL CALCULATE THE EQUATION FOR
THE "BEST" STRAIGHT LINE THROUGH A GIVEN SET OF DATA POINTS.

INPUT NUMBER OF DATA POINTS,
FOR THE STANDARD CURVE.
:6

INPUT (X,Y) FOR EACH DATA PT.
WHERE X IS THE CONCENTRATION OF THE STANDARD,
AND Y IS THE % TRANSMISSION.

:1.E-7,:14.0,    A =  .854

:50,:19.5,    A =  .710

:100,:29.0,    A =  .538

:150,:43.5,    A =  .362

:200,:63.0,    A =  .201

:250,:85.8,    A =  .067

COEFFICIENT OF DETERMINATION =    .998

INTERCEPT =   .858

SLOPE =  -.003


DO YOU WISH TO CALCULATE THE CONCENTRATIONS OF
GLUCOSE FOR EACH SERUM SAMPLE?

TYPE "1" FOR YES, OR ZERO, "0", FOR NO.
:1


INPUT NUMBER OF SAMPLES TO BE TESTED.
:10
```

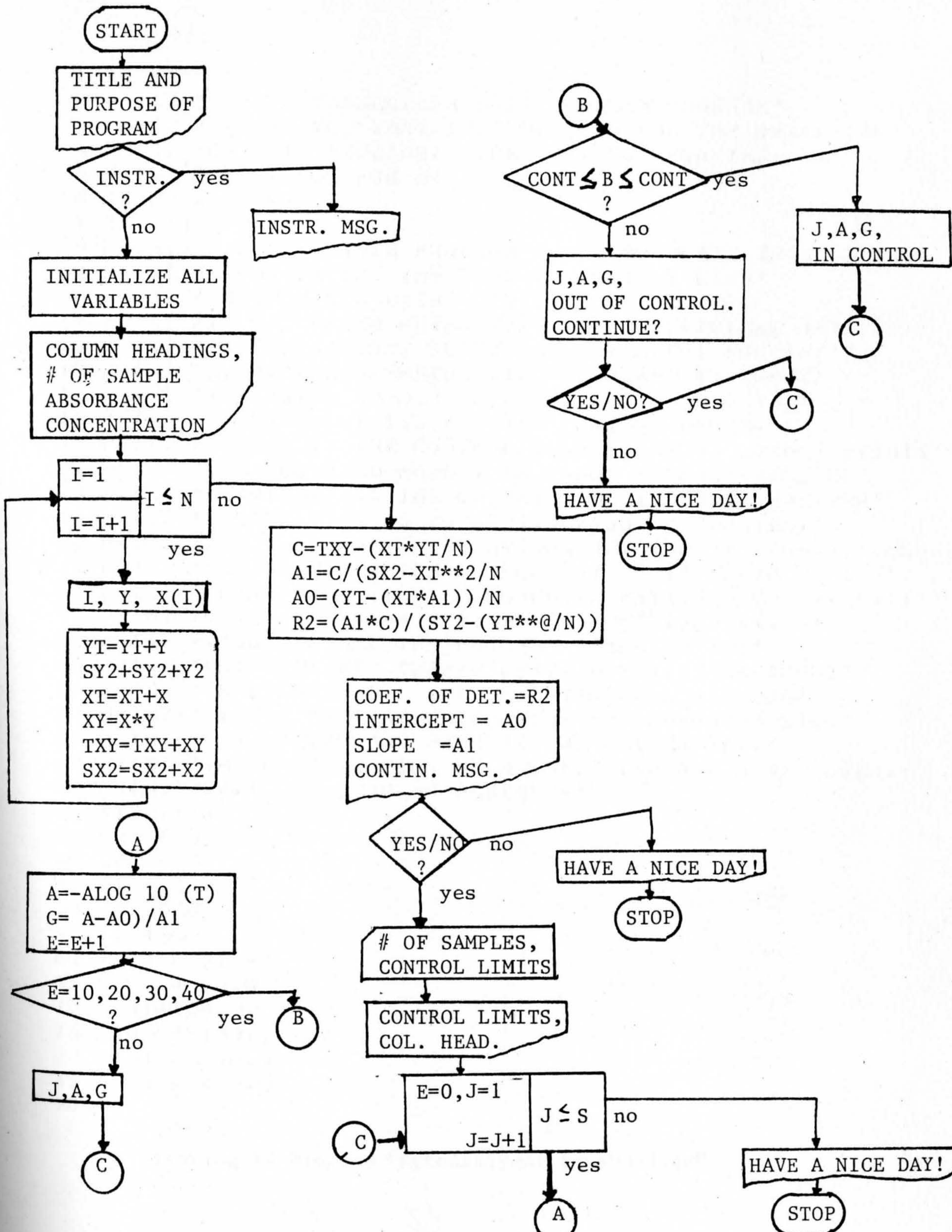
Listing 14. Program "Linear Least Squares" Run

```
INPUT %TRANSMISSION FOR EACH OF THE 10 SAMPLES,
ONE AT A TIME.

SAMPLE #  1
:14.0,    A =  .854

SAMPLE #  2
:19.5,    A =  .710

SAMPLE #  3
:29.0,    A =  .538

SAMPLE #  4
:43.5,    A =  .362

SAMPLE #  5
:63.0,    A =  .201

SAMPLE #  6
:85.8,    A =  .067

SAMPLE #  7
:27.0,    A =  .569

SAMPLE #  8
:17.0,    A =  .770

SAMPLE #  9
:30.0,    A =  .523

SAMPLE # 10
:50.0,    A =  .301


  SAMPLE        GLUCOSE
  NUMBER     CONCENTRATION
   1.00          1.26
   2.00         45.91
   3.00         99.38
   4.00        154.01
   5.00        203.91
   6.00        245.53
   7.00         89.75
   8.00         27.42
   9.00        103.95
  10.00        172.77


HAVE A NICE DAY!

>
```

Listing 14. Program "Linear Least Squares" Run (cont.)

```
K (    0)       11.00
> (    0)        9.00
G (   10)      172.77
G (    9)      103.95
G (    8)       27.42
G (    7)       89.75
G (    6)      245.53
G (    5)      203.91
G (    4)      154.01
G (    3)       99.38
G (    2)       45.91
G (    1)        1.26
G (    0)         .00
A (    0)         .30
P (    0)       50.00
J (    0)       11.00
> (    0)       14.00
S (    0)       10.00
> (    0)      200.00
> (    0)      190.00
M (    0)        1.00
> (    0)        6.00
R2(    0)        1.00
T2(    0)        7.45
A0(    0)         .86
A1(    0)        -.00
S2(    0)   562499.18
C (    0)     -141.02
X2(    0)    62499.96
XY(    0)       16.63
Y2(    0)         .00
> (    0)        3.00
> (    0)        5.00
> (    0)        2.00
Y (    0)         .07
T (    0)       85.80
X (    0)      250.00
> (    0)        1.00
I (    0)        7.00
N (    0)        6.00
YT(    0)        2.73
TX(    0)      200.25
XT(    0)      750.00
SY(    0)        1.70
> (    0)        0.00
SX(    0)   137499.92

>
```

Listing 15. Symbol Table, "Linear Least Squares" Program



Listing 16. Flowchart for "AutoAnalyzer Data Handler" Program

```
1 ERASE
2 PRINT '        AUTOANALYZER DATA HANDLER PROGRAM'
3 PRINT /,/,'IF YOU WANT INSTRUCTIONS FOR THE OPERATION'
4 PRINT 'OF THIS PROGRAM, TYPE ONE, "1" FOR YES, OR'
5 PRINT 'ZERO, "0" FOR NO.'
6 INPUT Z
7 IF (Z) 3,30,8
8 PRINT /,/,'        THIS PROGRAM WILL CALCULATE THE'
9 PRINT 'EQUATION FOR THE "BEST" STRAIGHT LINE'
10 PRINT 'THROUGH SIX DATA POINTS.'
11 PRINT /,'        AFTER OUTPUTTING THE COEFFICIENT OF'
12 PRINT 'DETERMINATION, SLOPE AND INTERCEPT FOR THE'
13 PRINT 'LINE, THE OPERATOR WILL BE ASKED TO INPUT'
14 PRINT 'THE TOTAL NUMBER OF SAMPLES AND ALSO THE'
15 PRINT 'ACCEPTED LIMITS FOR THE CONTROL VALUE.'
16 PRINT /,'        IF THE CONTROL VALUES ARE NO LONGER WITHIN'
17 PRINT 'THE ACCEPTED RANGE, AN ERROR MESSAGE WILL BE'
18 PRINT 'PRINTED, AND THE OPERATOR MUST DECIDE WHETHER'
19 PRINT 'CALCULATION IS TO CONTINUE OR BE ABORTED.'
20 PRINT /,'        DEPRESS THE RETURN KEY (UPPER RIGHT) FOLLOWING'
21 PRINT 'ALL INPUTS WHICH SHOULD NOT BE TYPED IN BY THE'
22 PRINT 'OPERATOR UNTIL THE COMPUTER PRINTS A COLON, (:).'
23 PRINT /,/,'        ***** RESTART INSTRUCTIONS *****'
24 PRINT /,'TO RESTART THE PROGRAM, DEPRESS THE'
25 PRINT 'ALT MODE KEY (UPPER LEFT) ANY TIME AN INPUT'
26 PRINT 'IS REQUESTED.  THE COMPUTER WILL RESPOND'
27 PRINT 'WITH A ">", AT WHICH TIME THE OPERATOR WILL'
28 PRINT 'TYPE "1.G" AND DEPRESS THE RETURN KEY.'
29 PRINT 'THE PROGRAM WILL BE REINITIATED AND START OVER.'
30 PRINT /,/,'        PROGRAM BEGINS >'
32 FISET(1)
35 SX2 = 0
40 SY2 = 0
45 XT = 0
50 TXY = 0
55 YT = 0
60 X(1) = 1.E-7
65 X(2) = 50.0
70 X(3)=100.0
75 X(4) = 150.0
80 X(5) = 200.0
85 X(6) = 250.0
90 N=6
```

Listing 17 Program "AutoAnalyzer DataHandler"

```
92 PT=GET(0)
93 PRINT /,/,'STANDARD #    ABSORBANCE    CONCENTRATION'
95 DO 160 I=1,N,1
100 T=(GET(I))/PT
104 Y=-ALOG 10 (T)
105 FMT (F9,3)
110 PRINT I,'   ',Y,'          ',X(I)
115 YT=YT+Y
120 Y2=Y**2
125 SY2=SY2+Y2
130 X=X(I)
135 XT=XT+X
140 XY=X*Y
145 TXY=TXY+XY
150 X2=X**2
155 SX2=SX2+X2
160 CONTINUE
165 C=TXY-(XT*YT/N)
170 S2X=XT**2
175 A1=C/(SX2-S2X/N)
180 A0=(YT-(XT*A1))/N
185 T2Y=YT**2
190 R2=(A1*C)/(SY2-(T2Y/N))
195 PRINT/,/,'COEFFICIENT OF DETERMINATION = ',R2
197 PRINT /,'INTERCEPT = ',A0
200 PRINT /,'SLOPE = ',A1
205 PRINT /,/,/,'TO CALCULATE THE CONCENTRATIONS OF'
210 PRINT 'GLUCOSE FOR EACH SERUM SAMPLE,'
215 PRINT 'TYPE ONE, "1", FOR YES OR ZERO, "0", FOR NO.'
220 INPUT M
225 IF (M) 215,460,230
230 PRINT /,/,'INPUT TOTAL NUMBER OF SAMPLES, INCLUDING THE 6'
235 PRINT 'STANDARDS, AND ALL CONTROLS (MAXIMUM OF 40).'
240 INPUT S
245 PRINT 'INPUT THE LIMITS OF THE CONTROL VALUE.'
250 PRINT 'HIGH'
255 INPUT HIGH
260 PRINT 'LOW'
265 INPUT LOW
270 PRINT /,/,'CONTROL LIMITS FOR GLUCOSE CONCENTRATION ARE '
275 PRINT LOW,' TO ',HIGH
280 PRINT /,/,/,'     SAMPLE         ABSORBANCE    GLUCOSE'
281 PRINT '      NUMBER                   CONCENTRATION'
290 E=0
295 DO 400 J=1,S,1
297 T=(GET(J))/PT
300 A=-ALOG 10 (T)
305 G=(A-A0)/A1
310 E=E+1
```

Listing 17. Program "AutoAnalyzer Data Handler" (cont.)

```
315 IF (10-E) 320,350,399
320 IF (20-E) 330,350,399
330 IF (30-E) 340,350,399
340 IF (40-E) 420,350,399
350 IF(G-HIGH) 360,360,375
360 IF(G-LOW) 375,365,365
365 PRINT ' ',J,'     ',A,'        ',G,'  IN CONTROL.'
370 GOTO 400
375 PRINT ' ',J,'     ',A,'        ',G,'  OUT OF CONTROL.  CONTINUE?'
376 INPUT O
378 IF (O) 400,420,400
399 PRINT ' ',J,'     ',A,'       ',G
400 CONTINUE
420 PRINT /,/,/,'          ***** HAVE A NICE DAY! *****'
430 STOP
>
```

Listing 17. Program "AutoAnalyzer Data Handler" (cont.)

```
>20000,0050D

20000   0231 0026 0036 0055 0103 0141 0203 0131
20010   0177 0115 0064 0060 0062 0151 0070 0104
20020   0071 0131 0156 0056 0070 0045 0072 0103
20030   0154 0043 0170 0173 0130 0061 0056 0050
20040   0104 0120 0103 0135 0071 0053 0177 0201
20050   0064

>
```

Listing 18. Internally Stored Data Points

```
AUTOANALYZER DATA HANDLER PROGRAM

IF YOU WANT INSTRUCTIONS FOR THE OPERATION
OF THIS PROGRAM, TYPE ONE, "1" FOR YES, OR
ZERO, "0" FOR NO.
:1


     THIS PROGRAM WILL CALCULATE THE
EQUATION FOR THE "BEST" STRAIGHT LINE
THROUGH SIX DATA POINTS.

     AFTER OUTPUTTING THE COEFFICIENT OF
DETERMINATION, SLOPE AND INTERCEPT FOR THE
LINE, THE OPERATOR WILL BE ASKED TO INPUT
THE TOTAL NUMBER OF SAMPLES AND ALSO THE
ACCEPTED LIMITS FOR THE CONTROL VALUE.

     IF THE CONTROL VALUES ARE NO LONGER WITHIN
THE ACCEPTED RANGE, AN ERROR MESSAGE WILL BE
PRINTED, AND THE OPERATOR MUST DECIDE WHETHER
CALCULATION IS TO CONTINUE OR BE ABORTED.

     DEPRESS THE RETURN KEY (UPPER RIGHT) FOLLOWING
ALL INPUTS WHICH SHOULD NOT BE TYPED IN BY THE
OPERATOR UNTIL THE COMPUTER PRINTS A COLON, (:).


     ***** RESTART INSTRUCTIONS *****

TO RESTART THE PROGRAM, DEPRESS THE
ALT MODE KEY (UPPER LEFT) ANY TIME AN INPUT
IS REQUESTED.  THE COMPUTER WILL RESPOND
WITH A ">", AT WHICH TIME THE OPERATOR WILL
TYPE "1.G" AND DEPRESS THE RETURN KEY.
THE PROGRAM WILL BE REINITIATED AND START OVER.

   PROGRAM BEGINS >
```

Listing 19. Program "AutoAnalyzer Data Handler" Run

```
STANDARD #     ABSORBANCE     CONCENTRATION
   1.000          .842               .000
   2.000          .708             50.000
   3.000          .531            100.000
   4.000          .359            150.000
   5.000          .198            200.000
   6.000          .067            250.000

COEFFICIENT OF DETERMINATION =        .998

INTERCEPT =          .849

SLOPE =        -.003


TO CALCULATE THE CONCENTRATIONS OF
GLUCOSE FOR EACH SERUM SAMPLE,
TYPE ONE, "1", FOR YES OR ZERO, "0", FOR NO.
:1


INPUT TOTAL NUMBER OF SAMPLES, INCLUDING THE 6
STANDARDS, AND ALL CONTROLS (MAXIMUM OF 40).
:40
INPUT THE LIMITS OF THE CONTROL VALUE.
HIGH
:130
LOW
:110


CONTROL LIMITS FOR GLUCOSE CONCENTRATION ARE
  110.000 TO    130.000
```

Listing 19. Program "AutoAnalyzer Data Handler" Run (cont.)

```
    SAMPLE          ABSORBANCE      GLUCOSE
    NUMBER                          CONCENTRATION
     1.000            .842             2.165
     2.000            .708            44.439
     3.000            .531            99.704
     4.000            .359           153.956
     5.000            .198           204.389
     6.000            .067           245.346
     7.000            .235           192.657
     8.000            .081           241.119
     9.000            .298           172.917
    10.000            .469           119.411   IN CONTROL.
    11.000            .503           108.501
    12.000            .486           114.065
    13.000            .164           215.191
    14.000            .437           129.512
    15.000            .352           155.975
    16.000            .429           131.924
    17.000            .235           192.657
    18.000            .143           221.532
    19.000            .522           102.700
    20.000            .437           129.512   IN CONTROL.
    21.000            .616            73.024
    22.000            .421           134.295
    23.000            .359           153.956
    24.000            .151           219.031
    25.000            .641            65.450
    26.000            .106           233.391
    27.000            .095           236.757
    28.000            .240           191.117
    29.000            .494           111.311
    30.000            .522           102.700   OUT OF CONTROL.   CONTINUE?
:1
    31.000            .583            83.650
    32.000            .352           155.975
    33.000            .282           178.126
    34.000            .359           153.956
    35.000            .216           198.649
    36.000            .429           131.924
    37.000            .551            93.508
    38.000            .081           241.119
    39.000            .074           243.249
    40.000            .469           119.411   IN CONTROL.


    ***** HAVE A NICE DAY! *****

>
```

Listing 19. Program "AutoAnalyzer Data Handler" Run (cont.)

```
>.L

> (     0)       12.000
> (     0)      420.000
> (     0)       40.000
O (     0)        1.000
> (     0)      340.000
> (     0)       30.000
> (     0)      330.000
> (     0)       20.000
> (     0)      400.000
> (     0)      365.000
> (     0)      375.000
> (     0)      360.000
> (     0)      350.000
> (     0)      399.000
> (     0)      320.000
> (     0)       10.000
G (     0)      119.411
A (     0)         .469
J (     0)       41.000
E (     0)       40.000
LO(     0)      110.000
HI(     0)      130.000
S (     0)       40.000
> (     0)      230.000
> (     0)      215.000
M (     0)        1.000
R2(     0)         .998
T2(     0)        7.319
A0(     0)         .849
A1(     0)        -.003
S2(     0)   562499.189
C (     0)     -139.401
X2(     0)    62499.966
XY(     0)       16.855
Y2(     0)         .005
> (     0)        9.000
Y (     0)         .067
T (     0)         .340
I (     0)        7.000
PT(     0)      153.000
N (     0)        6.000
X (     6)      250.000
> (     0)      250.000
> (     0)        6.000
X (     5)      200.000
```

Listing 20. Symbol Table for "AutoAnalyzer Data Handler" Program

```
> (    0)      200.000
> (    0)        5.000
X (    4)      150.000
> (    0)      150.000
> (    0)        4.000
X (    3)      100.000
> (    0)      100.000
X (    2)       50.000
> (    0)       50.000
> (    0)        2.000
X (    1)         .000
> (    0)         .000
X (    0)      250.000
YT(    0)        2.705
TX(    0)      198.758
XT(    0)      750.000
SY(    0)        1.665
> (    0)        0.000
SX(    0)   137499.928
> (    0)        1.000
> (    0)        8.000
> (    0)        3.000
Z (    0)        1.000

>
```

Listing 20. Symbol Table for "AutoAnalyzer Data Handler" Program (cont.)

```
LINEAR LEAST SQUARES REGRESSION

     THIS PROGRAM WILL CALCULATE THE EQUATION FOR
THE "BEST" STRAIGHT LINE THROUGH A GIVEN SET OF DATA POINTS.

INPUT NUMBER OF DATA POINTS,
FOR THE STANDARD CURVE.
:6

INPUT (X,Y) FOR EACH DATA PT.
WHERE X IS THE CONCENTRATION OF THE STANDARD,
AND Y IS THE % TRANSMISSION.

:1.E-7,:14.0,      A =   .854

:50.0,:19.5,      A =   .710

:100.0,:29.0,      A =   .533

:150.0,:43.5,      A =   .368

:200.0,:63.0,      A =   .201

:250.0,:85.5,      A =   .068

COEFFICIENT OF DETERMINATION =    .998

INTERCEPT =    .858

SLOPE =   -.003
```

Listing 21. Dry Run Simulation "Linear Least Squares" Program

```
DO YOU WISH TO CALCULATE THE CONCENTRATIONS OF
GLUCOSE FOR EACH SERUM SAMPLE?

TYPE "1" FOR YES, OR ZERO, "0", FOR NO.
:1


INPUT NUMBER OF SAMPLES TO BE TESTED.
:40

INPUT %TRANSMISSION FOR EACH OF THE 40 SAMPLES,
ONE AT A TIME.

SAMPLE #  1
:14.0,    A =  .854

SAMPLE #  2
:19.5,    A =  .710

SAMPLE #  3
:29.0,    A =  .538

SAMPLE #  4
:43.5,    A =  .362

SAMPLE #  5
:63.0,    A =  .201

SAMPLE #  6
:85.5,    A =  .068
```

Listing 21. Dry Run Simulation "Linear Least Squares" Program (cont.)

```
SAMPLE #  7
:58.0,    A =  .237

SAMPLE #  8
:82.5,    A =  .084

SAMPLE #  9
:49.0,    A =  .310

SAMPLE # 10
:34.0,    A =  .469

SAMPLE # 11
:31.5,    A =  .502

SAMPLE # 12
:33.0,    A =  .481

SAMPLE # 13
:68.0,    A =  .167

SAMPLE # 14
:33.0,    A =  .420

SAMPLE # 15
:44.3,    A =  .354

SAMPLE # 16
:37.2,    A =  .429

SAMPLE # 17
:58.0,    A =  .237

SAMPLE # 18
:71.5,    A =  .146

SAMPLE # 19
:30.0,    A =  .523

SAMPLE # 20
:36.5,    A =  .438

SAMPLE # 21
:24.0,    A =  .620

SAMPLE # 22
:37.5,    A =  .426

SAMPLE # 23
:43.7,    A =  .360
```

Listing 21. Dry Run Simulation "Linear Least Squares" Program (cont.)

```
SAMPLE # 24
:70.5,     A =  .152

SAMPLE # 25
:23.0,     A =  .638

SAMPLE # 26
:78.5,     A =  .105

SAMPLE # 27
:80.0,     A =  .097

SAMPLE # 28
:57.0,     A =  .344

SAMPLE # 29
:32.0,     A =  .495

SAMPLE # 30
:30.0,     A =  .523

SAMPLE # 31
:26.0,     A =  .585

SAMPLE # 32
:44.0,     A =  .357

SAMPLE # 33
:52.0,     A =  .284

SAMPLE # 34
:43.5,     A =  .352

SAMPLE # 35
:60.5,     A =  .213

SAMPLE # 36
:37.0,     A =  .432

SAMPLE # 37
:28.0,     A =  .553

SAMPLE # 38
:68.0,     A =  .081

SAMPLE # 39
:84.0,     A =  .076

SAMPLE # 40
:34.0,     A =  .469
```

Listing 21. Dry Run Simulation "Linear Least Squares" Program (cont.)

SAMPLE NUMBER	GLUCOSE CONCENTRATION
1.00	1.17
2.00	45.88
3.00	99.42
4.00	154.13
5.00	204.10
6.00	245.30
7.00	192.94
8.00	240.48
9.00	170.19
10.00	120.38
11.00	110.58
12.00	116.86
13.00	214.40
14.00	135.89
15.00	156.59
16.00	133.02
17.00	192.94
18.00	221.17
19.00	104.00
20.00	130.46
21.00	73.69
22.00	134.10
23.00	154.75
24.00	219.27
25.00	58.15
26.00	233.76
27.00	236.33
28.00	190.60
29.00	112.71
30.00	104.00
31.00	84.69
32.00	155.67
33.00	178.21
34.00	154.13
35.00	198.54
36.00	132.29
37.00	94.69
38.00	241.30
39.00	242.91
40.00	120.38

HAVE A NICE DAY!

>

Listing 21. Dry Run Simulation "Linear Least Squares" Program (cont.)

```
>  (    0)     10.000
K  (    0)     41.000
>  (    0)      9.000
G  (   40)    120.885
G  (   39)    248.912
G  (   38)    241.296
G  (   37)     94.690
G  (   36)    132.293
G  (   35)    198.635
G  (   34)    154.128
G  (   33)    178.209
G  (   32)    155.670
G  (   31)     84.691
G  (   30)    103.998
G  (   29)    112.705
G  (   28)    190.595
G  (   27)    236.329
G  (   26)    233.775
G  (   25)     68.150
G  (   24)    219.273
G  (   23)    154.747
G  (   22)    134.104
G  (   21)     75.892
G  (   20)    130.457
G  (   19)    103.998
G  (   18)    221.174
G  (   17)    198.942
G  (   16)    133.020
G  (   15)    156.587
G  (   14)    135.891
G  (   13)    214.402
G  (   12)    116.857
G  (   11)    110.581
G  (   10)    120.885
G  (    9)    179.191
G  (    8)    240.480
G  (    7)    198.948
G  (    6)    245.299
G  (    5)    204.098
G  (    4)    154.128
G  (    3)     99.484
G  (    2)     45.878
G  (    1)      1.172
G  (    0)       .000
```

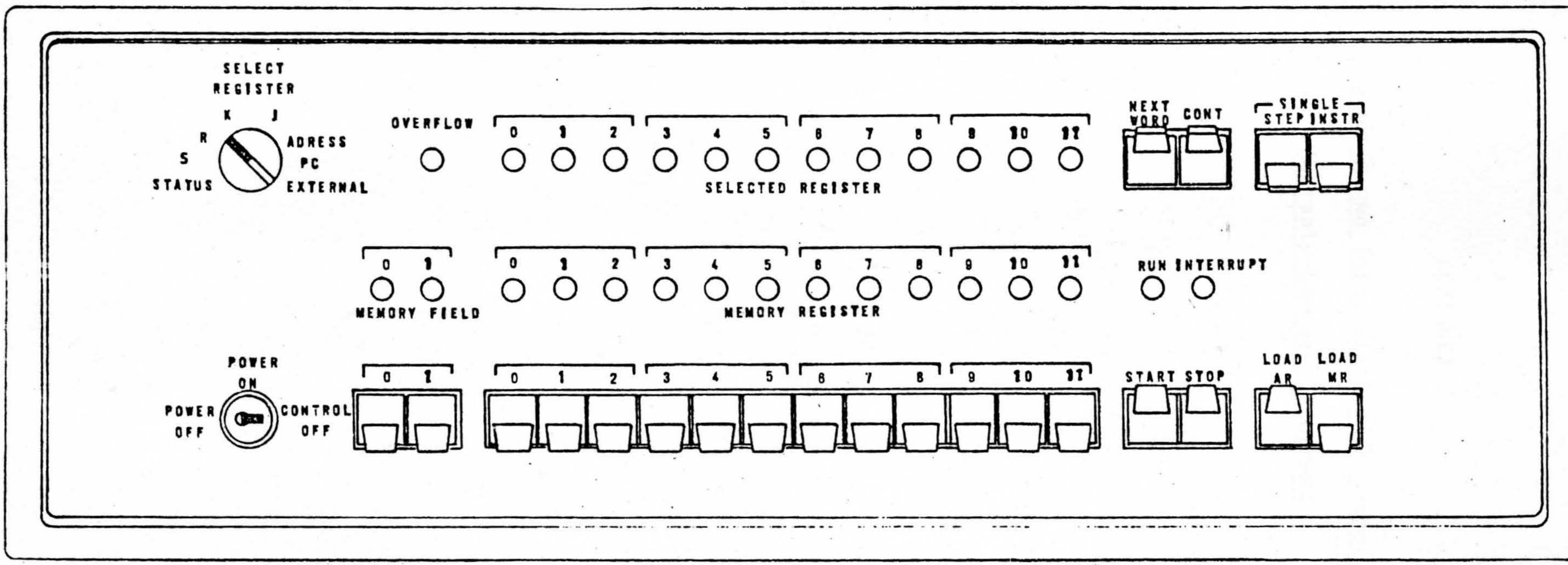
Listing 22. Symbol Table for Dry Run Simulation

```
A (   0)        .469
P (   0)      34.000
J (   0)      41.000
> (   0)      14.000
S (   0)      40.000
> (   0)     200.000
> (   0)     190.000
M (   0)       1.000
> (   0)       6.000
R2(   0)        .998
T2(   0)       7.462
A0(   0)        .858
A1(   0)       -.003
S2(   0) 562499.189
C (   0)    -140.830
X2(   0)  62499.966
XY(   0)      17.008
Y2(   0)        .005
> (   0)       3.000
> (   0)       5.000
> (   0)       2.000
Y (   0)        .068
T (   0)      35.500
X (   0)     250.000
> (   0)       1.000
I (   0)       7.000
N (   0)       6.000
YT(   0)       9.732
TX(   0)     200.625
XT(   0)     750.000
SY(   0)       1.698
> (   0)       0.000
SX(   0) 137499.923

>
```

Listing 22. Symbol Table for Dry Run Simulation (cont.)

APPENDIX B

ND 812 Central Processor Controls and Indicators



ND 812 Central Processor Controls and Indicators

APPENDIX C

Loading and Initialization Procedure

Assembly-Language Programs

LOADING AND INITIALIZATION PROCEDURE

ASSEMBLY-LANGUAGE PROGRAMS

The following procedures describe loading and initializing for the object tapes of any Assembly-Language program via teletype. It is assumed that the ND 812 computer and the teletype are turned on and operating properly before the procedures are performed.

a. Depress ND 812 STOP switch.

b. Set Teletype START/FREE/STOP switch to FREE.

c. Load Object Tape into Teletype reader. Advance the tape so that the leader (level 8 punched) is over the read head.

d. Set Teletype START/FREE/STOP switch to START.

e. Simultaneously depress ND 812 LOAD AR and NEXT WORD switches. The tape is automatically fed through the Teletype reader. When tape motion stops, set ND 812 SELECT REGISTER switch to J. All ND 812 front-panel SELECTED REGISTER lamps should be off. If any lamps are on, repeat steps a through e.

f. If all lamps are off, set ND 812 SWITCH REGISTER switches to the correct starting address of the program. Then, in sequence, depress ND 812 LOAD AR and START switches.

g. If the program does not start, repeat all steps.

APPENDIX D

Loading and Initialization Procedure

Nutran Intrepreter and Nutran Programs

LOADING AND INITIALIZATION PROCEDURE

NUTRAN INTERPRETER AND NUTRAN PROGRAMS

The following procedures describe loading and initializing for the Nutran Interpreter via Teletype. It is assumed that the ND 812 Computer and the Teletype are turned on and operating properly before the procedures are performed.

a. Depress ND 812 STOP switch.

b. Set Teletype START/FREE/STOP switch to FREE.

c. Load Nutran Interpreter Program (41-0059) tape, Part I into Teletype reader. Advance the tape so that the leader (level 8 punched) is over the read head.

d. Set Teletype START/FREE/STOP switch to START.

e. Simultaneously depress ND 812 LOAD AR and NEXT WORD switches. The tape is automatically fed through the Teletype reader. When tape motion stops, set ND 812 SELECT REGISTER switch to J. All ND 812 front-panel SELECTED REGISTER lamps should be off. If any lamps are on, repeat steps a through e.

f. If all lamps are off, set Teletype START/FREE/STOP switch to FREE and remove tape. Load Nutran Interpreter tape, Part II, as described in step c. (Repeat with parts III and IV.)

g. Repeat step d.

h. Repeat step e. If any lamps are on, repeat steps f through h.

i. If all lamps are off, Set ND 812 SWITCH REGISTER switches to 0100_8 (switch 5 up, all others down). Then, in sequence, depress ND 812 LOAD AR and START switches. The Teletype responds by typing . If > is not typed, repeat all steps.

j. Type ".E" followed by a carriage return.

k. Load Nutran Language program into Teletype reader.

l. Set Teletype START/FREE/STOP switch to START.

m. At the end of the tape, type a carriage return, and ".nG", where n is the first line of the program.

APPENDIX E

Glossary

GLOSSARY[62]

Absolute address -- 1. An actual location in storage of a particular unit of data; address that the control unit can interpret directly. 2. The label assigned by the engineer to a particular storage area in the computer. 3. A pattern of characters that identifies a unique storage location or device without further modification.

Accumulator (AC) -- The accumulator is a 4, 8, 12 or 16-bit register that functions as a holding register for arithmetic, logical, and input-output operations. Data words may be fetched from memory to the AC or from the AC into memory. Arithmetic and logical operations involve two operands, one held in the AC, the other fetched from memory. The result of an operation is retained in the AC. the AC may be cleared, complemented, tested, incremented or rotated under program control. The AC also serves as an input-output register. Programmed data transfers pass through the AC.

A/D analog-digital converter -- Circuit used to convert information in analog form into digital form (or vice versa), e.g., in a digital voltmeter, and other devices.

Address, symbolic -- Arbitrary identification of a particular word, function, or other information without regard to the location of the information.

Algorithm -- A prescribed set of well-defined rules or processes for the solution of a problem in a finite number of steps, for example, a full statement of an arithmetic procedure for evaluating sin x to a stated precision.

American Standard Code for Information Interchange (ASCII) -- Usually pronounced "ASKEE." A standard data-transmission code that was introduced to achieve compatibility between data devices. It consists of 7 information bits and 1 parity bit for error checking purposes, thus allowing 128 code combinations. If the eighth bit is not used for parity, 256 code combinations are possible.

Analog -- In electronic computers, the term refers to a physical system in which the performance of measurements yields information concerning a class of mathematical problems.

Assemble -- 1. In digital computer programming, to put together subprograms which finally make up a complete program. To perform sone or all of the following functions: 2. Translation of symbolic operation codes into machine codes. 3. Allocation of storage, to the extent at least of assigning storage locations to successive instructions. 4. Computation of absolute or relocatable addresses from symbolic addresses. 5. Insertion of library routines. 6. Generation of sequences of symbolic instructions by the insertion of specific parameters into macro instructions.

GLOSSARY (CONT.)

Assembler -- The essential capability of a assembler is to translate symbolically represented instructions into their binary equivalents. A well designed computer is reflected in a versatile, efficient assembly language instruction set. It is a computer program which operates on symbolic input data to produce from such data machine instructions by carrying out such functions as, translation of symbolic operation codes into computer operating instructions, assigning locations in storage for successive instructions, or computation of absolute addresses from symbolic notation.

Automatic interrupt -- 1. Interruption caused by program insturction as contained in some executive routine; interruption not caused by programmer, but due to engineering of devices. 2. An automatic program-controlled interrupt system that causes a hardware jump to a predetermined location.

Binary -- A characteristic, property, or condition in which there are but two possible alternatives, e.g., the binary number system using 2 as its base and using only the digits zero (0) and one (1).

Binary code -- A code in which every code element is one of two distinct types of values. For example, the presence or absence of a pulse.

Binary digit (bit) -- 1. A numeral in the binary scale of notation. This digit may be zero (0) or one (1). It may be equivalent to an on or off condition, a yes, or a no. Often abbreviated to (bit). 2. The kind of number that computers use internally. There are only two binary digits, 1 and 0, otherwise known as "on" and "off".

Carry-over effects -- The effect of mixing of samples which occurs as different samples pass through a system. The first portions of the second sample serve to wash out the trailing portions of the first sample. The effects of this mixing can be observed in the output of the recorder.

Central processing unit (CPU) -- 1. A unit of a computer that includes the circuits controlling the interpretation and execution of instructions. Synonymous with mainframe. Abbreviated CPU. 2. The central processor of a computer system contains main storage, arithmetic unit, control registers, and scratchpad memory.

Clear -- An activity to place one or more storage locations into a prescribed state, usually zero or the space character.

GLOSSARY (CONT.)

Compatible -- Refers to the capability of direct interconnection; usually implies no requirement for code, speed, or signal level conversion.

Complement -- A number which is derived form the finite positional notation of another by one of the following rules: True complement --Subtract each digit form 1 less than the base; then add 1 to the lease significant digit and execute all required carries. Base minus 1's complement -- Subtract each digit from 1 less than the base (e.g., 9's a complement in base 10, 1's complement in the base 2, etc.

Computer run -- 1. Refers to the processing of a batch of transactions while under the control of one or more programs, and against all the files that are affected to produce the required output. 2. Performance of one routine, or several routines automatically linked so that they form an operating unit, during which manual manipulations are not required of the computer operator.

Computer word -- Relates to that sequence of bits or characters treated as a unit and capable of being stored in one computer location.

Conversational language -- Refers to various languages that utilize a near-English character set which facilitates communication between the computer and the user.

Data, analog -- A physical representation of information such that the representation bears an exact relationship to the original information.

Digital -- Refers to the use of discrete integral numbers in a given base to represent all the quantities that occur in a problem or calculation. It is possible to express in digital form all information stored, transferred, processed, or transmitted by a dual-state condition (i.e., on-off, true-false, and open-closed.)

Editor -- An editor is a general-purpose text editing program used to prepare source program tapes. Original text entered via the teletypewriter and held in memory may be changed and corrected. The user can insert, delete or change lines of text, insert, delete and change characters within a line without retyping the line, locate lines containing key words and list or punch any portion of the text. Some editors require a minimum of 4K of RAM memory and are supplied on a binary tape with a User's Manual.

Enabled -- Refers to a state of the central processing unit that allows the occurrence of certain types of interruptions.

Entry point -- Refers to various specific locations in a program segment which other segments can reference.

GLOSSARY (CONT.)

Flag -- Refers to a bit (or bits) used to store one bit of information. A flag has two stable states and is the software analogy of a flip-flop.

Flag bit -- Refers to a specific information bit that indicates a type or form of demarcation that has been reached. This may be carry, overflow, etc. Generally the flag bit refers to special conditions such as various types of interrupts.

Flowchart -- 1. Usually a programmers tool for determining a sequence of operations as charted using sets of symbols, directional marks, and other representations to indicate stepped procedures of computer operation. Flowcharts also enable the designer to conce½tualize the procedure necessary and to visualize each step and item on a program. A complete flowchart is often a necessity to the achievement of accurate final code. 2. A graphical representation for the definition, analysis, or solution of a problem, in which symbols are used to represent operations, data, flow, equipment, etc. 3. A flowchart represents the path of data through a problem solution. It defines the major phases of the processing as well as the various data media used.

Gate -- A circuit having one output and several inputs, the output remaining unenergized until certain input conditions have been met. When used in conjunction with computers, a gate is also called an AND circuit. A gate can also be a signal to trigger the passage of other signals through a circuit.

High order -- Pertaining to the weight or significance assigned to the digits of a number, e.g., in the number 123456, the highest order digit is one, the lowest order digit is six. One may refer to the three high order bits of a binary word, as another example.

Increment --A software operation most oftem associated with stacks and stack pointers. Bytes of information are stored in the stack register at the addresses contained in the stack pointer. The stack pointer is decremented after each byte of information is entered into the stack; it is incremented after each byte is removed from the stack. Increment also refers to various addressable registers.

Initialize --1. Refers to a program or hardware circuit which will return a program, a system or a hardware device to an original state. 2. To set an instruction, counter, switch, or address to a specified starting condition at a specified time in a program.

Input -- 1. An adjective referring to a device or collective set of devices used for bringing data into another device. 2. A channel

GLOSSARY (CONT.)

for impressing a state on a device or logic element. 3. Pertaining to a device, process, or channel involved in an input process or to the data or states involved in an input process. In the English language, the adjective "input" may be used in place of "input data," "input signal," "input terminal," etc. when such usage is clear in a given context. 4. Pertaining to a device, process, or channel involved in the insertion of data or states, or to the data or states involved. 5. One, or a sequence of, input states.

Interface -- Refers to instruments, devices or a concept of a common boundary or matching of adjacent components, ciucuits, equipment, or system elements. An interface enables devices to yield and/or acquire information from one device or program to another. Although the terms adapter, handshake, buffer have similar meaning, interface is more distinctly a connection to complete an operation.

Interpreters -- There are occasions when neither assembly language or a compiler language is adequate. It will take too long to write and debug short programs whenever users want to run another trial-and-error calculation. An interpretive language fills this need. An interpreter is a program which operates directly on a source program in memory. The interpreter translates the instructions of the source program one by one and executes them immediately. It is not common practice to use interpreters to translate and execute source programs, since they are slow, but they have several advantages on specific problems.

Interrupt -- Various interrupts relate to the suspension of normal operations or programming routines of microprocessors and are most often designed to handle sudden requests for service or change. As peripheral devices interface with CPU's, various interrupts occur on frequent bases. Multiple interrupt requests require the processor to delay or prevent further interrupts; to break into a procedure; to modify operations, etc. and after completion of the interrupt task, to resume the operation from the point of interrupt.

Jump -- The Jump instruction or operation, like the Branch instruction, is designed to control the transfer of operations from one point to another point in a control or applications program. Jumps differ from Branches by avoiding the use of the Relative Addressing Mode.

Keyboard -- Keyboards fall into three basic types -- alphanumeric, numeric variety and mixed. Alphanumeric keyboards are used for word processing and teleprocessing. Numeric only keyboards are used on touch-tone telephones, accounting machines and calculators. The touch-tone telephone has come into significant use as a calculator and data input and voice output device.

Leader -- An unused or blank length of tape at the beginning of a reel of tape preceding the start of the recorded data.

GLOSSARY (CONT.)

Listing, assembly -- Refers to a printed list which is the by-product of an assembly procedure. It lists in logical instruction sequence all details of a routine showing the coded and symbolic notation next to the actual notation established by the assembly procedure. This listing is highle useful in the debugging of a routine.

Low order -- That which pertains to the weight or significance assigned to the digits farthest to the right within a number; e.g., in the number 123456, the low order digit is six. One may refer to the three low-order bits of a binary word as another example.

Machine language -- The final language all computers must use is binary. All other programming languages must be compiled or translated ultimately into binary code before entering the processor. Binary language is machine language.

Memory, main -- Usually the CPU or fastest storage device of a computer and the one from which instructions are executed.

Minicomputer -- 1. Generally a minicomputer is a mainframe that sells for less than $25,000. Usually it is a parallel binary system with 8, 12, 16, 18, 24 or 36-bit word length incorporating semiconductor or magnetic core memory offering from 4K words to 64K words of storage and a cycle time of 0.2 to 8 microseconds or less. A bare minicomputer (one without cabinet, console and power supplies) consisting of a single PC card can sell for less than $1,000 in OEM quantities. 2. These units are characterized by higher performance than microcomputers or programmable calculators, richer instruction sets, higher price and a proliferation of high level languages, operating systems, and networking methodologies.

Mnemonic -- Refers to a technique to assist a programmer's memory. Mnemonics is the art of imporving the efficiency of the memory in computer storage.

Mnemonic code -- Often referred to as 'memory codes' these are designed to assist programmers to remember instructions corresponding to a given operation. MPY for multiply, for example. Source statements can be written in this symbolic language and then translated into machine language.

Object program --1. A source program that has been automatically translated into machine language. 2. The final or 'target' program is referred to as the object program. The source program is developed first, and this is translated into the machine program for internal computer operation. 3. A set of machine language instructions for the solution of a problem, obtained as the end result of a compilation ½rocess. It is generated from the source program. 4. The absolute coding output by a processor program.

GLOSSARY (CONT.)

Octal -- A numbering system basic to computer operation. A positional notation system using 8 as a base, instead of 2, as in binary, 10, as in decimal, etc.

Octal digits -- the symbols 0, 1, 2, 3, 4, 5, 6, and 7 used in the octal numbering system.

Off-line -- Refers to equipment or devices not under direct control of the central processing unit. Also concerns description terminal equipment not connected to a transmission line.

On-line -- Relates to equipment, devices, or systems in direct interactive communication with the central processing unit. May also be used to describe terminal equipment connected to a transmission line.

Operand -- The fundamental quantity on which a mathematical operation is performed. Usually a statement consists of an operator and an operand. The operator may indicate an 'add' instruction; the operand thus will indicate what is to be added.

Operation -- Generally refers to the action specified by a single computer instruction or pseudo-instruction.

Output -- Refers to information and data transferred from the internal storage of a computer to output devices or external storage.

Paper tape -- Refers to strips of paper capable of storing or recording information. Storage may be in the form of punched holes, partially punched holes, carbonization or chemical change of impregnated material, or by imprinting. Some paper tapes, such as punched paper tapes, are capable of being read by the input device of a computer or a transmitting device by sensing the pattern of holes which represent coded information.

Peripheral equipment -- Units which work in conjunction with a computer but are not a part of it (e.g., tape reader, analog-to-digital converter, typewrite, etc.)

Program -- 1. A set of instructions arranged in a proper sequence for directing a digital computer in performing a desired operation or operations. 2. To prepare a program. 3. A sequence of audio signals transmitted for entertainment or information. 4. The basic computer preparation procedure. 5. A plan for the solution of a problem.

Pulse -- 1. The variation of a quantity having a normally constant value. This variation is characterized by a rise and a decay of a finite duration. 2. An abrupt change in voltage, either positive or negative, which conveys information to a circuit.

GLOSSARY (CONT.)

Reader --Refers to a device capable of sensing information stored in an off-line memory media (cards, paper tape, magnetic tape) and generating equivalent information in an on-line memory device (register, memory locations).

Read-in -- 1. Refers to the act of placing data in storage at a specified address. 2. To sense information contained in some source and transmit this information to an internal storage.

Record -- Refers to a collection of fields; the information relating to one are a activity in a data processing activity; sometimes called item.

Register -- A digital-computer device capable of retaining information, often that contained in a small subset (e.g., one word) of the aggregate information. A temporary storage device used for one or more words to facilitate arithmetical, logical or transferral operation.

Reliability -- The probability that a device will perform adequately for the length of time intended and under the operating environment encountered.

Restore -- Refers to return of a cycle index, a variable address, or other computer word to its initial value.

Return address -- Relates to that part of a subprogram that connects it with the main program.

Semantic error -- 1. One which results in ambiguoug or erroneous meaning of a computer program. Most programs have to be debugged to eliminate these errors before use.

Sequential operation -- The performance of actions one after the other in time. The actions referred to are of a large scale as opposed to the smaller scale operations referred to by the term serial operation.

Skip -- Refers to an instruction to proceed to the next instruction, a blank instruction.

Software -- Software items are programs, languages, and procedures of a computer system.

Source program -- A program that can be translated automatically into machine language. It thereby becomes an object program.

GLOSSARY (CONT.)

Teletype, ASR33 -- Various teletype models may be purchased through suppliers with microcomputers. A typical unit has a pedestal base, and includes the page copy teleprinter, keyboard assembly, paper tape perforator with chad box. Optional equipment includes modern couplers, printer disable kits, and other items. Purchase of a complete system through a specific supplier assures complete hardward compatibility.

Text editor -- A text editor provides the system user with a convenient and flexible source text generation system. Source statements are entered via any source input device/file. The entered source text may be output, statements added, deleted or modified. The text editor permits the order of statements or groups of statements to be altered at any time. The final text is output to a source device/file for use as input to an Assembler.

Throughtput -- Relates to the speed with which problems, programs, or segments are performed. Throughput can vary from application as well as from one piece of equipment to another although they are the same brand, and even model.

Toggle switch -- A two-position snap switch operated by a projecting lever to open or close circuits.

Trap -- A selective circuit that attenuates undesired signals but does not affect the desired ones. An unprogrammed conditional jump to a known location, automatically activated by hardware, with the location from which the jump occurred.

Turn around time -- Time elapsing between the beginning of a process, and the end of the process. In regards to computers, the time between the input of data and the receipt of hard-copy output by the operator.

Two's complement -- In some systems the ALU performs standard binary addition using the 2's complement numbering system to represent both positive and negative numbers. The positive numbers in 2's complement representation are identical to the positive numbers in standard binary. However, the negative 2's complement is the reverse of the negative standard binary plus 1.

Word -- A group of characters occupying one storage location in a computer. It is treated by the computer circuits as an entity, by the control unit as an instruction, and by the arithmetic unit as a quantity.

REFERENCES

1. Westinghouse Electric Corp., Lab. Man., (January), 1972, p. 18.

2. M. C. Farquhar and D. E. Hirch, Amer. Lab., (January), 1973, pp. 51, 52.

3. Westinghouse, *op. cit.*

4. *Ibid.*, p. 20.

5. K. M. Aldous, D. G. Mitchell, and J. S. Merritt, Amer. Lab., 8, (September), 1976, p. 34.

6. A. E. Rappoport, W. D. Gennaro, and W. J. Constandse, Hosp. Prog. 50, 114, (1969).

7. *Ibid.*

8. A. E. Rappoport and W. D. Gennaro, Computers in Biomedical Research, Vol. IV, 215, (1974).

9. W. N. Shannon, Lab Man., (march), 1972, p. 33.

10. Westinghouse, *op. cit.*, p. 26.

11. W. N. Shannon, *op. cit.*,p. 26.

12. *Ibid.*

13. A. E. Rappoport, W. D. Gennaro, and W. J. Constandse, *op. cit.*

14. *Ibid.*

15. W. N. Shannon, *op. cit.*, p. 27.

16. J. F. Barkley and P. S. Shoenfield, Amer. Lab., 7, (February), 1975, p. 19.

17. A. L. Louderback, Amer. Lab., (February), 1974, p. 128.

18. S. Raymond, Lab. Man., 13, (May), 1975, p. 32.

19. M. C. Farquhar, *op. cit.*, p. 52.

20. A. E. Rappoport and E. N. Rappoport, J. Amer. Hosp. Assn., 44, 114, 1970.

21. *Ibid.*

REFERENCES (CONT.)

22. Westinghouse, *op. cit.*, p. 20.

23. *Ibid.*

24. Digital Computers in Scientific Instrumentation, Applications to Chemistry, S. P. Perone and D. O. Jones, McGraw-Hill Book co., New York, 1973, p. 6.

25. *Ibid.*

26. *Ibid.*, pp. 6, 7.

27. *Ibid.*

28. *Ibid.*

29. *Ibid.*, pp. 7 - 9.

30. J. W. Frazer, Amer. Lab., (February), 1973, p. 21.

31. Perone, *op. cit.*, pp. 10, 11.

32. Microcomputer, Dictionary and Guide, C. J. Sippl andD. A. Kidd, Matrix Publishers, Inc., Illinois, 1975, p. 285.

33. Perone, *op. cit.*, p. 2.

34. *Ibid.*

35. Computer Fundamentals for Chemists, J. S. Mattson, H. B. Mark, Jr., and H. C. MacDonald, Jr., ed.'s, Marcel Dekker, Inc., New York, 1973, p. 6.

36. Analytical Chemistry, 2nd. Ed., G. D. Christian, John Wiley & Sons, New York, 1977, p. 552.

37. Practical Automation for the Clinical Laboratory, 2nd. ed., W. L. White, M. M. Erikson, and S. C. Stevens, C. V. Mosby Comp., St. Louis, 1972, p. 224.

38. *Ibid.*, p. 259.

39. *Ibid.*, pp. 276, 277.

40. *Ibid.*, p. 279.

41. HP·25 Applications Programs, 00025-90011, RV.C 8/75, Hewlett-Packard Co., 1975, p. 88.

42. *Ibid.*

43. *Ibid.*

REFERENCES (CONT.)

44. Technical Specifications, ND560 ADC Analog to Digital Converter, Nuclear Data Inc., Illinois.

45. Analog-Digital Designer, Model EU-801A, Owner's Manual, Heath Company, Michigan, 1968, p. 3.

46. Model SM-102A Programmable Timer, Operation/Service Manual, Heath Company, Michigan, 1973, p. 1-1.

47. Voltage Reference Source, (Zener Stabilized), Model EU-80A, Owner's Manual, Heath Company, Michigan, p. 3.

48. Instruction Manual, Model 142, HF VCG Generator, Wavetek, Indiana, p. 1-1.

49. Principles of Programming the ND812 Computer in Assembly Language, IM41-0000-02, Nuclear Data, Inc., Illinois, 1974,p. 3-4.

50. *Ibid.*, p. 3-5.

51. *Ibid.*, p. 3-6.

52. *Ibid.*, p. 3-7.

53. *Ibid.*

54. *Ibid.*, p. 3-8.

55. *Ibid.*

56. *Ibid.*, p. 3-9.

57. *Ibid.*, pp. 5-21, 5-22.

58. *Ibid.*, pp. 5-17, 7-18.

59. *Ibid.*, pp. 5-23, 5-24.

60. Perone, *op. cit.*, p. 294.

61. Nutran User and Programmers Guide, IM41-0059-02, Nuclear Data, Inc., Illinois, 1973, p. 1-4.

62. C. J. Sippl and D. A. Kidd, *op. cit.*

BIBLIOGRAPHY

A Guide for Software Documentation, D. Walsh, Advanced Computer Techniques Corporation, New York, 1969.

Amer. Lab., J. Vasily and T. H. Doyne, 27, (January), 1973.

Amer. Lab., L. J. Brocato, 47, (September), 1973.

Amer. Lab., D. C. Beckwith, 77, (February), 1973.

Amer. Lab., E. Irack, and H. V. Malmstadt, 56, (February), 1973.

Amer. Lab., A. D. Overstreet, D. G. Larson, P. R. Rony, 78, (February), 1974.

Amer. Lab., C. Spector, 39, (September), 1971.

Amer. Lab., D. G. Larsen and P. R. Rony, 57, (July), 1974.

Amer. Lab., S. W. Downer, 93, (February), 1974.

Amer. Lab., S. Ahuja, 12, (September), 1973.

Amer. Lab., D. F. Krawczyk and K. V. Byram, 55, (January), 1973.

Amer. Lab., 7(9), W. E. Goldstein, R. W. Rogers and M. A. Grenshaw, 47, (September), 1975.

Amer. Lab., H. S. Ames and R. W. Crawford, 37, (February), 1977.

Amer. Lab., 6(9), P. E. Bosley, 27, (September), 1974.

Amer. Lab., 8(2), G. M. Bobba, and L. F. Donaghey, 27, (February), 1976.

Amer. Lab., 4(9), J. C. Elkins, 37, (September), 1972.

Computing With Minicomputers, F. Gruenberger, D. Babcock, Melville Publishing Co., Los Angeles, California, 1973.

Hardware Instruction Manual, ND560 ADC Analog to Digital Converter, IM88-0415-01, Nuclear Data, Inc., Schaumburg, Illinois, 1973.

Hardware Instruction Manual, ND4410 Data Acquisition and Display System, IM88-0428-00, Nuclear Data, Inc., Palatine, Illinois, 1972.

Lab. Man., W. H. Shannon, 22, (February), 1972.

Lab. Man., 11(11), A. H. Free and H. M. Free, 25, (November), 1973.

BIBLIOGRAPHY (CONT.)

Lab. Man., 12(7), R. A. Richards, 26, (July), 1974.

Modern Hospital, A. E. Rapport, W. D. Gennaro, and W. J. Constandse, 94, (April), 1968.

Res. and Dev., 25(9), R. G. Helmke, 24, (september), 1974.

Software Instruction Manual, ND812 Symbolic Text Editor, IM41-0002-05, Nuclear Data, Inc., Schaumberg, Illinois, 1974.

Software Instruction Manual, BASC-12 General Assembler, 4K/8K, Teletype/ Line Printer, IM41-0001-01, Nuclear Data, Inc., Palatine, Illinois, 1973.

Software Instruction Manual, ND812 Utilities, IM41-0005-02, Nuclear Data, Inc., Schaumberg, Illinois, 1974.