DESIGN AND IMPLEMENTATION OF A LABORATORY COMPUTER NETWORK FOR DATA ACQUISITION AND ANALYSIS

A DISSERTATION

SUBMITTED TO THE FACULTY OF ATLANTA UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

BY

TERRY JEROME GREEN

DEPARTMENT OF CHEMISTRY

ATLANTA, GEORGIA

July 1984

R= ix T = 125

ABSTRACT

CHEMISTRY

GREEN, TERRY J. B.S., SOUTH CAROLINA STATE COLLEGE, 1976
M.S., ATLANTA UNIVERSITY, 1980

DESIGN AND IMPLEMENTATION OF A LABORATORY COMPUTER NETWORK FOR DATA ACQUISITION AND ANALYSIS

Advisor: Professor G. Scott Owen

Dissertation date July 1984

A Laboratory Data System (LDS) was implememented in the Chemistry Research Laboratory. The design of the system was a distributed local area network (LAN) of microcomputers. This LAN is based on a combination of the Corvus Constellation and OMNINET hardware and software. The LAN has a Bus topology with all attached computers having access to a 20 Mbyte Winchester Disk Drive. There are a total of seven computers currently in the LAN - three APPLE II systems, two LSI-11 systems, one LSI-11/23 system, and one IBM PC system. Four spectrophotometers were interfaced to the LSI-11/23 computer via a 12-bit Analog Input/Output System (16 channels analog-to-digital converter and 2 channels digital-to-analog converter, DAC) and a real time clock (RTC). The instruments were a Varian Model 3700 Gas

Chromatograph, a Cary 17 UV-VIS-NIR Spectrophotometer, a Beckman 4240 Infrared (IR) Spectrophotometer, and a Durrum-Jasco J-20 Circular Dichroism (CD) Spectrophotometer. The two DAC channels were interfaced to an oscilloscope for real time graphics output and to a x-y plotter for hard copy graphics output. Subroutines which control the function of the two interface boards are written in MACRO-11 assembly language. Data aquisition programs were written in Fortran. Programs have been written for signal averaging and spectral smoothing. A sophisticated graphics program (AGRAPH) plots the data on a graphics terminal and a digital plotter.

Furthermore, the LAN has a powerful intercomputer communication protocol allowing data collected from the instruments to be stored as a part of a created database.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere appreciation to the faculty and staff of the chemistry department for my professional training. Sincere thanks also goes to Dr. G. Scott Owen for his professional guidance and advice during the creation of this project.

I am grateful to Mr. Armen S. Karapetian, an electrical engineer with Lockheed-Georgia Company, for his technical assistance and for designing special interfaces.

I would also like to thank Dr. James O. Currie of Pacific University for his help in setting up the OMNINET.

Finally, a special consideration to my parents and family for their encouragement, moral support and understanding through the years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure		Page

I. INTRODUCTION

The introduction of the digital computer as a new laboratory tool has brought about a significant improvement in analytical techniques. In recent years, laboratory automation has revolutionized the methods of laboratory analysis. Experiments are not only being automated but the use of computers makes possible experiments and methods that previously could not be done. An example of this would be an experiment in which the signal to noise ratio (S/N) must be enhanced in order to obtain reasonable results.

Because of available computer processing capabilities (either built-in or on-line), new types of instruments are being developed. In addition, the power of existing instruments is greatly enhanced by the utilization of both on-line and off-line computer processing. The enormous saving in data analysis time and the accompanying reduction in errors is dramatically changing much of experimental methodology.

In 1962, the first general purpose computer designed specifically for the laboratory was developed by Digital Equipment Corporation (DEC). This was the Laboratory Instrument Computer (LINC) which was the prototype for most laboratory computers. The LINC was designed to be flexible enough for connection to a variety of digital and analog signals originating from laboratory instruments. Still, because of the high cost of computers it was not always economical to computerize a laboratory. But as technology improved, bringing the cost of computers downward, it became more feasible to automate the research laboratory. Moreover, instruments are now being designed for use on-line to experimentation and processes. Thus, with the recent advances both in instrumentation and in computer hardware and software, computers will enhance the incorporation of analytical instrumentation into the experiment or process[1].

Chemical research will take increasing advantage of these developments. Some of the benefits to be derived are as follows: improved analytical accuracy, experimentation that otherwise would be difficult to perform, unattended operation, on-line graphics, and enough data to numerically model the processes being studied. Another benefit is the development of more innovative applications of laboratory data analysis techniques. Some of these applications include optimization of experiments based on data collected during an experimental run, simultaneous measurement and collection of different experimental variables and their combined use to obtain new types of data, and the control of large, complex experiments based on data collected and analyzed in an interconnected set of analytical procedures. To be able to perform such experimentation one needs to properly specify and design the apparatus to take full advantage of the computer(s) and instrumentation.

Automated systems can generate the vast quantity of data required to characterize complex systems. Interpretation of these large volumes of data by conventional techniques is very time consuming and perhaps impossible. However, the use of computer graphics can greatly aid in data reduction, interpretation and development of numerical models. Furthermore, incorporating computers and graphics on-line to chemical experiments and processes opens new opportunities for the study and control of complex systems. Systems having many variables can be characterized even when the variable interactions are nonlinear and the system cannot be represented by numerical methods and models. That is, large sets of data can be rapidly acquired, then modeling and graphic techniques can be used to obtain a partial interpretation plus the design of further experiments.

To obtain the maximum throughput from laboratory automation, each system in the laboratory should be doing an independent task. Also, all the computer systems should be connected with each other into an interactive network in which intercommunication is fully developed. Networks allow each system to communicate with each other thus permitting resource and data sharing.

The purpose of this research project was to implement a

Laboratory Data System which would be interfaced to the chemistry research instrumentation via a local area network for data acquisition and analysis. The following chapters will discuss the background of a Laboratory Data System including types of computer networks, data handling techniques, laboratory data management systems, the local area network design that was implemented, and the chemical instruments which were interfaced to the network.

II. BACKGROUND

A. Computer Networks

Although computers have increased the throughput of the research laboratory, they are also creating such volumes of data that in many places the chemist has become a high priced key operator, transcribing information from one computer to another. This problem can be alleviated by tying all of the computers in the laboratory into an interactive network in which intercommunication is fully developed.

A network can be defined as a collection of wires and electronics that links a group of computers and peripherals. Each network is composed of three parts: the computers or peripherals in the network, interface units, and the communications cables or wires that connect the pieces together. Networking permits the users to communicate, exchange information and share larger, more reliable peripherals such as hard-disk storage units, high-speed printers, digital plotters, and sophisticated display terminals. The cost and performance of a network are determined by two key factors: the network's topology (physical configuration) and its transmission medium (the cables or wires that link the various devices).

There are three basic network configurations: star, ring and bus. The majority of the early networks had the star topology. Many of the current networks have either ring or bus configurations. Both the ring and bus configurations are distributed and decentralized access topologies while the star is a centralized topology.

Star. In the star configuration (Figure 1) the nodes radiate from a common controller. This central controller contains an interface card for each node and a processor that manages systems communications. A typical star network can usually handle between 8 and 25 computers, but often several complete networks are linked to increase the total number of stations on the system. One advantage of the star networks is that each node on the network does not require its own interface, since the software protocol for network access is centralized in the controller processor. However, a failure at the central controller brings the entire network down.

Ring. In the ring configuration, the nodes are attached to each other, forming a ring (Figure 2). Messages are passed unidirectionally from node to node through some form of repeater until they reach their destination. The most common flow control and access strategy used for inserting and removing messages from the ring network is the control token concept[2]. In this method, a token packet (a unique 8-

bit pattern such such as 11111111) is passed around the ring. In order for a node to transmit, it must remove the token from the ring. Because no other nodes can then seize the token, they must remain idle while the node with the token transmits. After it stops transmitting, it returns the token to the ring, which passes it along to the next node. Clearly, a failure at a single node can bring the entire network down.

Bus. Nodes in the bus network have their receiver attached to a single cable that travels from node to node (Figure 3). The cable is either a single coaxial cable or a twisted-pair wire. Only the sending node has its transmitter attached to the bus and all nodes receive all messages essentially simultaneously. A common software protocol which allows access to the network while preventing (or at least minimizing) the chances of data collision is the carrier-sense multiple access with collision detection protocol (CSMA/CD)[2]. This requires each node to monitor the network and it begins to transmit only if no other node has begun transmitting at the same time. A problem using the CSMA/CD method is that an individual node might never gain access to the network in a situation characterized by heavy use.

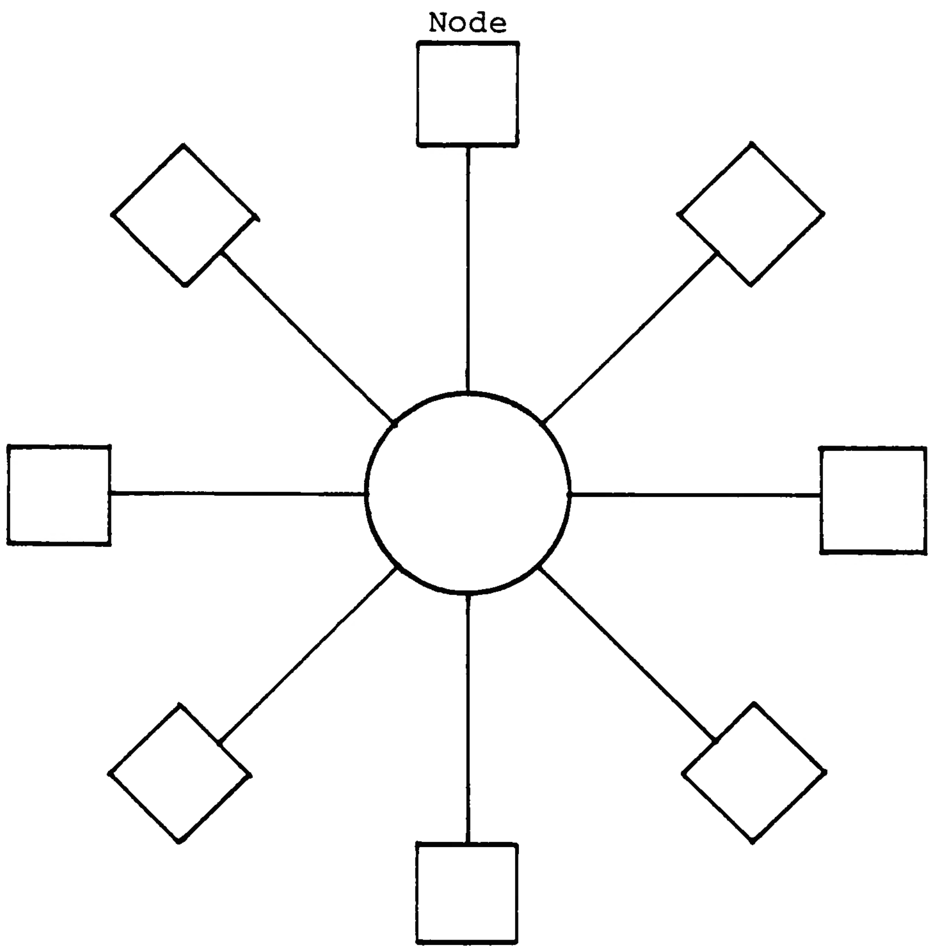


Figure 1. Star network.

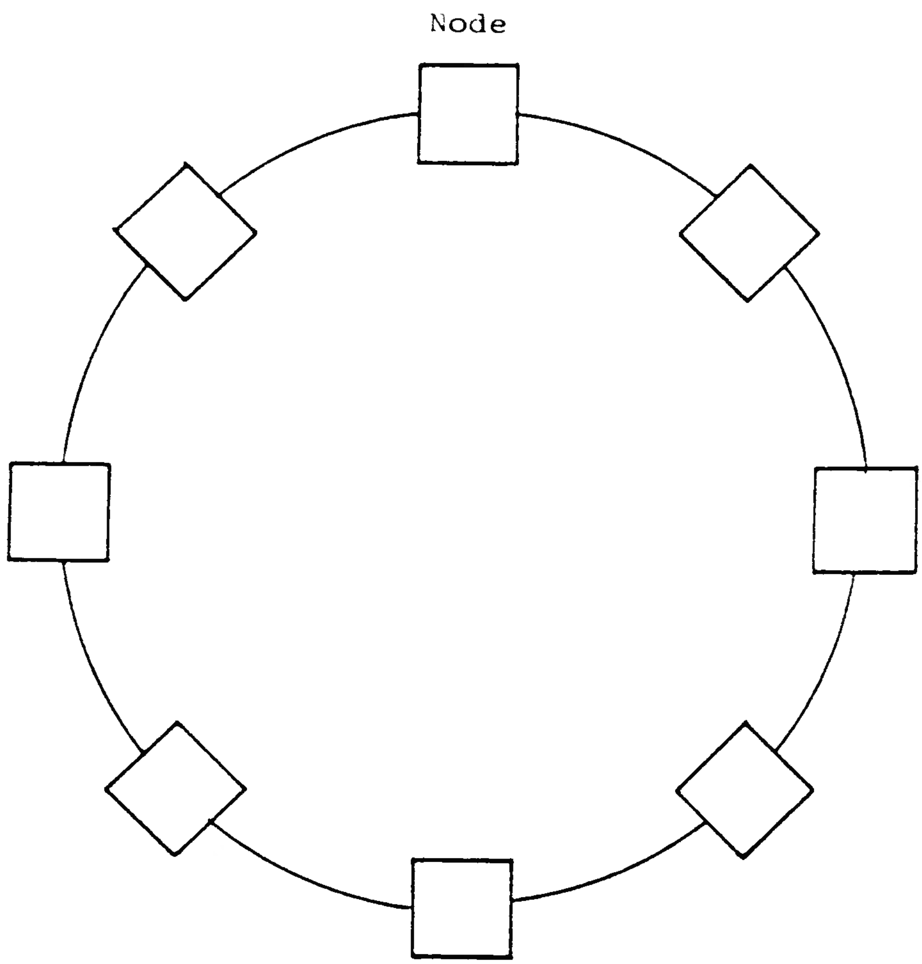


Figure 2. Ring network.

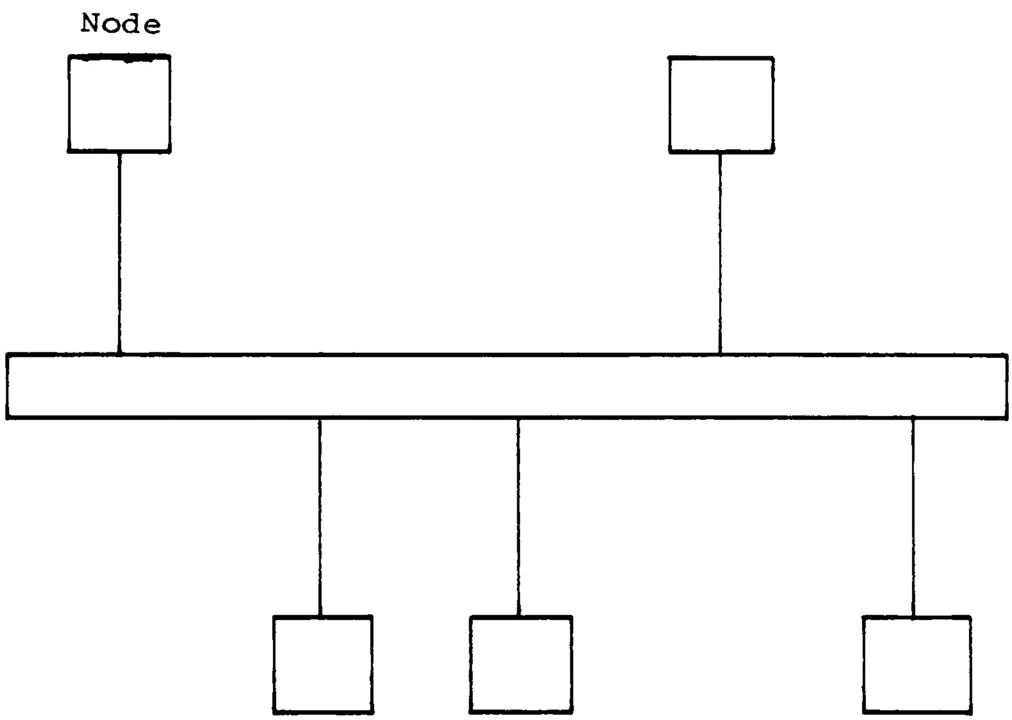


Figure 3. Bus network.

1. Local Area Networks

Networks that are generally characterized by inexpensive transmission media within a limited area, ranging from as much as six miles down to less than six tenths of a mile, and with data rates between 0.1 to 10 M bits/s can be defined as local area networks (LAN). With a LAN, it's possible to link personal computers and peripherals within an area the size of most company or department research laboratories, such as the LAN in the Quality Assurance Laboratory at Mary Kay Cosmetics[3], Figure 4.

2. Back-end Storage Networks

A subnetwork of a local computer network that allows several computers to share a common data storage medium is defined as a back-end storage network (BSN). This type of network is ideal for the microcomputer since it gives a relatively low cost microcomputer direct access to a large shared data medium. For example, BSNs provide an efficient means of interconnecting specialized data base computers to off-load data management tasks from a host computer. In addition, the back-end storage network was developed to handle the rapidly changing technologies of devices and systems, the increasing performance requirements, the increased complication and sophistication of interconnections

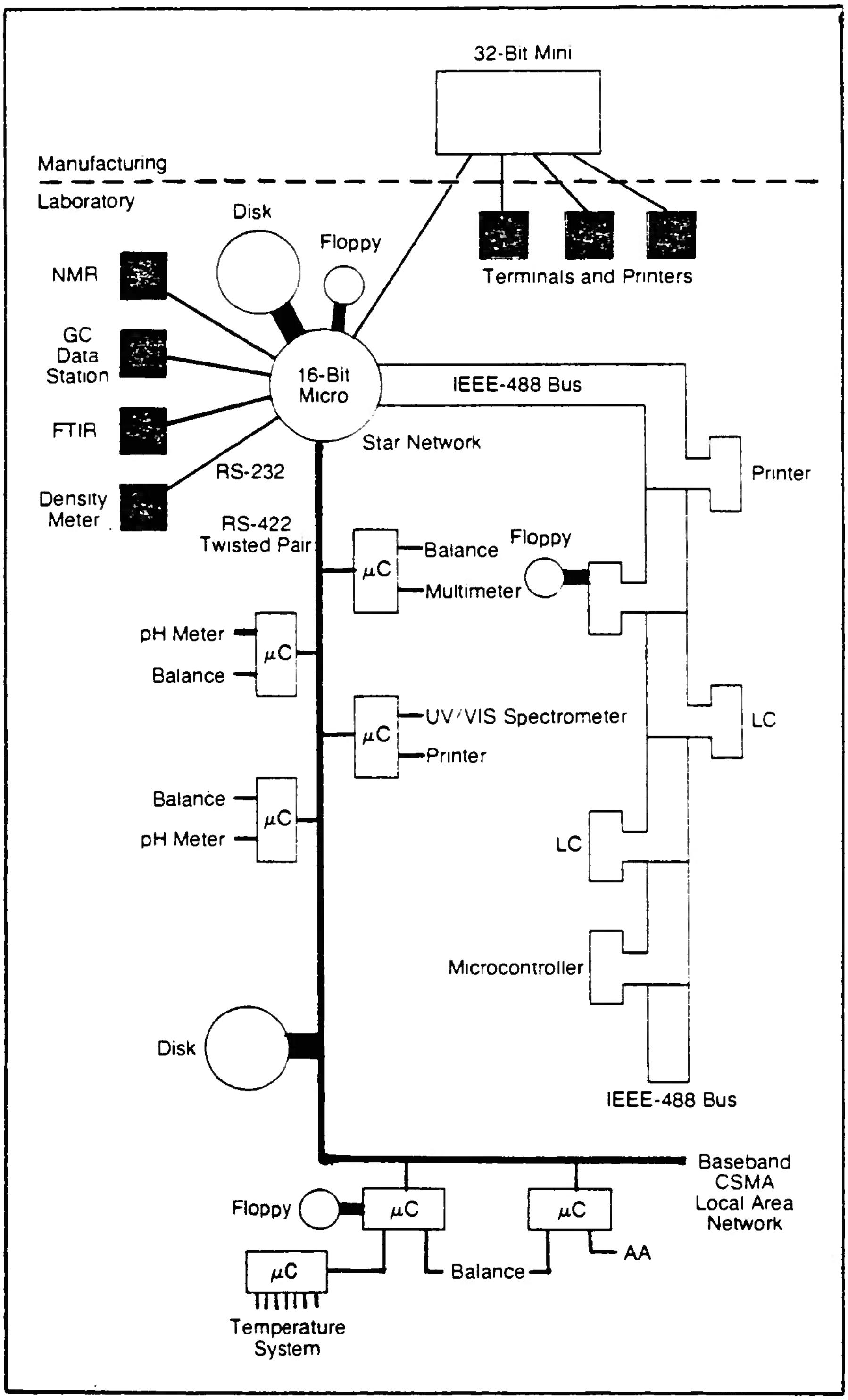


Figure 4. A bus network coupled to a star network.

and control, and the constant demand for improved reliability and availability in computer systems[4].

Two of the earlier BSNs that laid the ground work for the BSN architecture design were the Octopus and Skylab networks. The Octopus system was developed at Lawrence Livermore Laboratory. The other system, Skylab, was built by NASA for the Mission Control Center in Houston, Texas, to support satellites and their associated ground processing.

Octopus. The Octopus, a local area computer network at Lawrence Livermore Laboratory, was originally conceived as a star topology with a central switching node and a set of links to interconnect resources. The central node not only performed a packet switching function, it also handled terminal control and shared device control. This wide range of demands placed a major operating burden on the central node. Consequently, the Octopus was redesigned into various subnetworks of computers doing a particular task such as file transport and mass storage (the back-end networks). The file transport subnetwork was controlled by a PDP-10 computer, and the mass storage subsystem was controlled by a CDC-38500 computer system[5].

Skylab. Data transmitted from the orbiting satellite were received at remote tracking sites and then relayed to mission control center. At Houston center, a Univac 494 front-end processor received and routed the data to one or more of five IBM SYSTEM 1360 Model 75 computers. The data to be stored were then transmitted to the CDC Cyber 73 operating as a back-end processor[5].

B. Data Handling

Before the digitized representation of an analog signal can be stored in memory, some consideration must be given to the shape of the waveform and its frequency components. These factors will determine the minimum rate at which the analog signal can be digitized without loss of information. Also, the bulk of the data recorded in the laboratory is in the form of amplitude versus time. If the original signal is characterized in the form of amplitude versus frequency, which can be done by the application of a Fast Fourier Transform (FFT) to the digitized data, then this frequency spectrum provides insight into the noise content of the signal and the necessary sampling rate.

1. Data Acquisition

Data acquisition is the extraction of information from the instrument and the conversion of that information into computer compatible data (digitized data). The rate at which this data is acquired is a major consideration. In order to record the proper signal shape, the sampling frequency must be several times the signal frequency. An incorrect choice of sampling frequency can cause the actual signal to appear to be of lower frequency than the actual signal. This is known as aliasing. Aliasing occurs whenever the sampling rate is less than twice the maximum signal frequency. The frequency at which aliasing begins to occur is called the Nyquist frequency and as stated above, it is equal to one-half the sampling rate[6]. Although sampling at twice the maximum signal frequency should allow proper digitization, sampling should be done at least ten times the maximum signal frequency in order to obtain a good representation of the input signal.

Noise is another parameter that must be considered in the determination of the best data acquisition approach. Noise is any component of a signal which impedes observation, detection, or utilization of the information contained in the signal. Usually, noise contained in the signal is at a different frequency from the information contained in the signal. It is this property that allows noise to be partially removed or filtered from the analog signal.

Noise which is inherent in the system under study is endogenous noise. Thermal noise (Johnson noise) is caused by the random motion of electrons in detectors due to thermal agitation. Shot noise is the result of statistical fluctuations of charge carriers across a junction. Thermal noise and shot noise are random in origin and their measurement over a long period of time will trend to zero. Exogenous noise is created by interactions of the signal with elements external to the system. Examples include electromagnetic interference and radio frequency interference which are caused by induced currents in conductors from nearby radiating sources.

Various filtering techniques, whether analog or digital, can be applied to minimize the noise contained in the analog signal and to maximize the signal to noise (S/N) ratio in order to facilitate accurate and sensitive measurements. Analog filters are designed using resistors and capacitors (RC) to pass only signals with a frequency less than the specified RC cut-off frequency. Digital filters are software algorithms used to eliminate noise and enhance the S/N ratio on digitized data. There are several software algorithms used to accomplish this including boxcar averaging, ensemble averaging, the Savitzky-Golay weighted filter, and the FFT digital filter.

Boxcar Averaging. This technique can be applied where the rate of change of the analog signal is slow with respect to the sampling rate of the analog to digital converter (ADC). The signal is sampled several times at each particular point and averaged to replace that point. Figure 5 illustrates this technique.

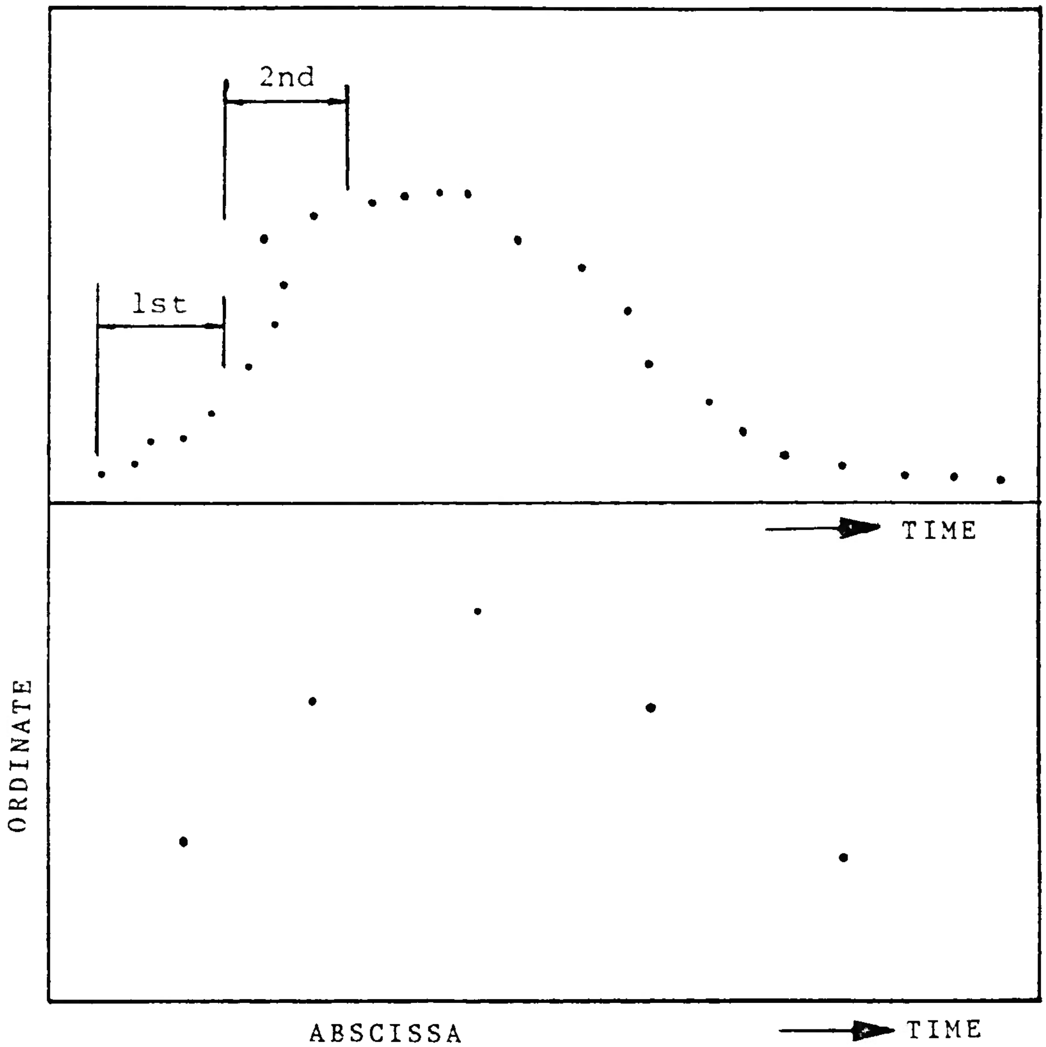


Figure 5. Boxcar averaging.

Since boxcar averaging is amenable to slowly changing signals it is often performed in real-time. This means that the data for each boxcar are sampled and averaged before the next boxcar. The boxcar algorithm must operate within the real-time environment. For example, suppose a time period of 0.1s is needed to measure an occurence in a particular sample. To accurately sample this occurence would require a sampling rate of 1.0KHz for a wide safety margin. This would be equivalent to a boxcar average every millisecond. A typical ADC takes about 20-25 microseconds to perform a conversion. If the necessary software to perform a boxcar (to make the conversions, calculate a sum, average, and store the data in memory) takes about 0.1 msec for each point, then an eight point boxcar will require about 0.8 msec leaving only 0.2 msec of computer time before the next boxcar is started. It is obvious that the disadvantage of this technique is the sampling rate. Thus, there is a trade-off between size of the boxcar and the number of data points per occurrence.

Boxcar averaging will only remove the high frequency noise components of a signal. The S/N enhancement is proportional to the square root of the number of points in the boxcar.

Ensemble Averaging. Ensemble averaging involves collecting successive sets of data from a repeated experiment and then normalizing the data by dividing the sum for each datum by the number of scans made as shown in Figure 6. The assumption is that the amplitude and phase of the noise relative to the data should be random so that the noise will average to zero. This technique filters out both low frequency and high frequency noise components. The S/N enhancement is proportional to the square root of the number of scans. The disadvantage of ensemble averaging is the extended amount of time required for many repetive scans.

Savitzky-Golay Weighted Filter. Savitzky and Golay[7] introduced a technique that emulates a least squares polynomial mathematical procedure for the manipulation of the raw or preprocessed data. In order to implement this technique certain conditions must be met:

1) data points occur at fixed uniform intervals on a chosen abscissa;

2) data points must be a continuous function.

In the calculations the ordinates of a fixed number of data points are multiplied by a convolution constant. The resulting products are summed and normalized to obtain the average convoluted ordinate at the central abscissa. The point at one end is then dropped, the next point at the other end is added, and the process is repeated. Figure 7 illustrares this process for a five point moving window. The set of numbers at the right are ordinate values, those

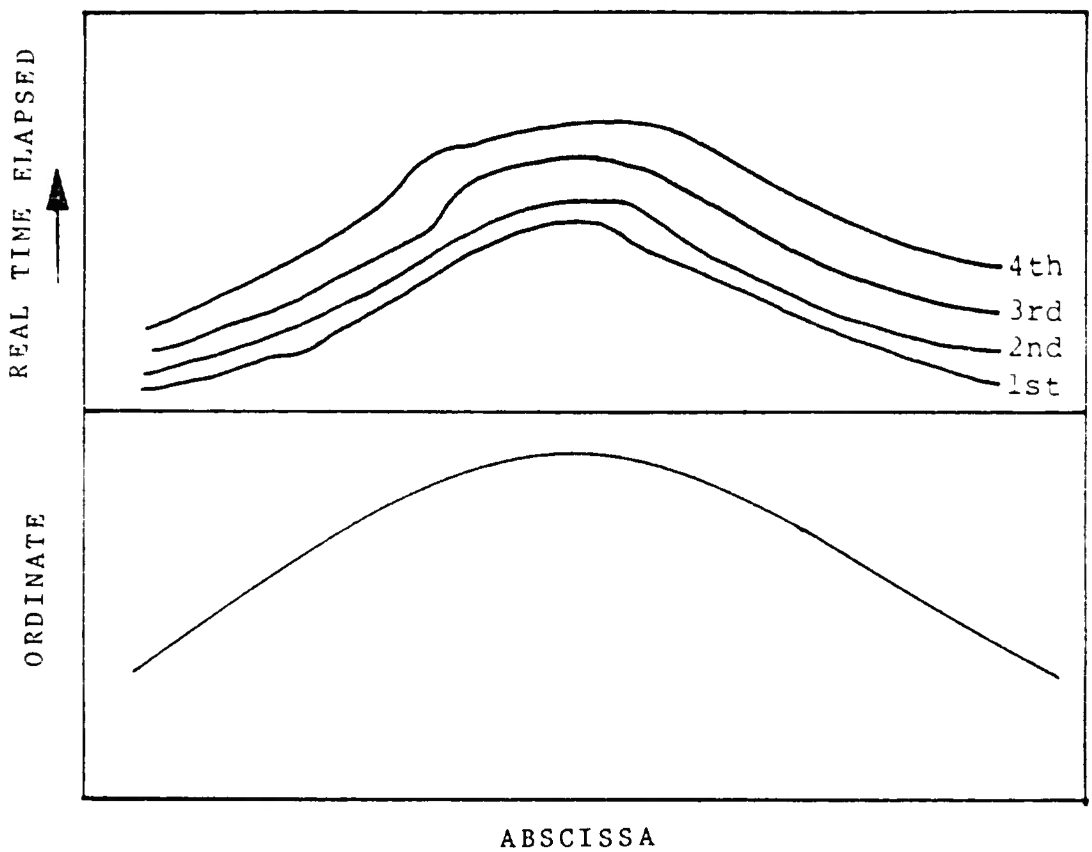


Figure 6. Ensemble averaging.

abscissa			ordinate
1800.0			705
1799.8			712
1799.6			717
1799.4			718
1799.2			721
1799.0	x_{-2}	c_{-2}	722
1798.0	x_{-1}	c_{-1}	725
1798.6	x	c	735
1798.4	x_{+1}	c_{+1}	736
1798.2	x_{+2}	c_{+2}	741
1798.0			746
1797.8			750

Figure 7. Implementation of a five point weighted filter on a set of data.

at the left are the abscissa values. The box in the center contains a number of convoluting integers (C_{-n}, C_{-n+1}, ... C_{n-1}, C_n) opposite to data reference points (X_{-n}, X_{-n+1}, ... X_{n-1}, X_n).

To convolute a particular datum each ordinate value is multiplied by its appropriate convolution integer, the resulting products are added, and finally divided by the sum of the convoluting integers. To obtain the next point in the moving average, the center block is moved down one location and the process is repeated. Table I shows the convoluting integers for cubic and quadratic functions for 5 to 25 points. These convoluting integers were chosen by Savitsky and Golay such that the above simple process is mathematically equivalent to a complete polynomial least squares analysis.

The S/N ratio increases with the square root of the number of points used in the convolution set. However, there are disadvantages in using too many points since peak distortion can be introduced. Best results are obtained by digitizing at high densities (short sampling times) and also by ensuring that no more than one inflection point is included in the convolution operation.

Fast Fourier Transform Digital Filter. The initial step in the FFT filtering process involves the computation of the FFT of the noisy time domain spectrum (Figure 8) to obtain

Table I. [a]The Weighting Functions for Cubic and Quadratic Functions for 5 to 25 Points

Convolutes Smoothing Quadratic Cubic A20 A30										
POINTS	25	23	21	19	17	15	13	11	9	7	5
-12	-253										
-11	-138	-42									
-10	-33	-21	-171								
-09	62	-2	-76	-136							
-08	147	15	9	-51	-21						
-07	222	30	84	24	-6	-78					
-06	287	43	149	89	7	-13	-11				
-05	322	54	204	144	18	42	0	-36			
-04	387	63	249	189	27	87	9	9	-21		
-03	422	70	284	224	34	122	16	44	14	-2	
-02	447	75	309	249	39	147	21	69	39	3	-3
-01	462	78	324	264	42	162	24	84	54	6	12
00	467	79	329	269	43	167	25	89	59	7	17
01	462	78	324	264	42	162	24	84	54	6	12
02	447	75	309	249	39	147	21	69	39	3	-3
03	422	70	284	224	34	122	16	44	14	-2	
04	387	63	249	189	27	87	9	9	-21		
05	322	54	204	144	18	42	0	-36			
06	287	43	149	89	7	-13	-11				
07	222	30	84	24	-6	-78					
08	147	15	9	-51	-21						
09	62	-2	-76	-136							
10	-33	-21	-171								
11	-138	-42									
12	-253										
NORM	5175	8059	3059	2261	323	1105	143	429	231	21	35

[a]Binkley, D. P.; Dessy, R. E. J. Amer. Chem. Soc., 1979, 56, 152.

the frequency domain spectrum (Figure 9). If it is assumed that the noise occurs at different frequencies than the information contained in the signal then the unwanted frequencies (noise) can be removed from the signal. The noise is removed by inputting the appropriate cut-off frequency while in the frequency domain. The inverse FFT is then performed to obtain the filtered spectrum (Figure 10).

2. Data Analysis

Various software algorithms have been developed to aid in the interpretation of the data generated by automated systems. Computer graphics are very useful in characterizing complex systems. Plotting routines not only provide analytical information but also provide visual aids crucial to the understanding of the system under investigation. In addition, plotting algorithms can aid in the development and the optimization of analytical methods.

Computational algorithms such as least-squares analysis, correlation functions, and deconvolution techniques can be used to numerically characterize the system. Furthermore, the FFT method has been applied in the analysis of a number of analytical techniques in the chemical laboratory in recent years. A few of these techniques include multiplex gas chromatography[8], linear parameter estimation of fused

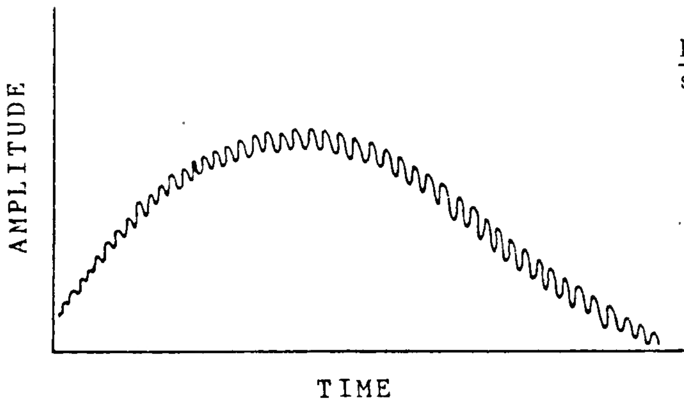


Figure 8. Noisy Gaussian spectrum.

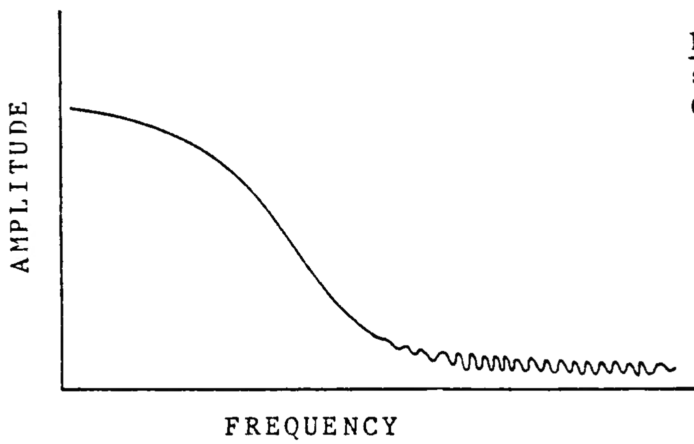


Figure 9. Frequency spectrum of noisy Gaussian peak.

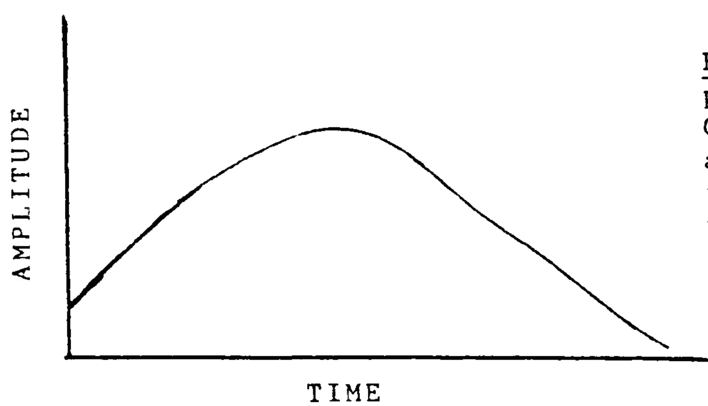


Figure 10. Inverse FFT performed on the noisy Gaussian peak with the appropriate cut-off frequency to obtain the filtered spectrum.

peak systems, and the interpolation of sampled data obtained by chromatographic, spectroscopic, and electrochemical techniques[9,10].

C. Laboratory Data Management System

When storing data generated by laboratory automation as a part of a created database for comparison with previous experiments or for further analysis, some sort of Laboratory Data Management System (LDMS) is required. The LDMS is used to keep track of the experiments and results. It can do this by keeping separate records for each analysis type, with each record consisting of several fields of information. For instance, there could be fields for the investigator's name, the date, and the instrument type for a particular sample. The LDMS would control the access to the database. Passwords or locks on data structures can easily define no access, read only, or read/write access for individual users. In addition, an LDMS is used in preparing reports based on the experiments.

Many techniques have been developed for the LDMS to store and retrieve information from the storage medium:

Sequential Access. This consists of the key words (investigator name, project number, etc.) ordered alphabetically as in a dictionary. There are two commonly used sequential access methods that help to expedite entry into the proper record: partitioned sequential access methods (PSAM) and indexed sequential access methods (ISAM). They allow the usage of either sample number or date to locate the pertinent data rapidly.

Ordered Index Access. This involves structuring the data base in a more complex way at the outset. The investigator names are kept in an alphabetized index. Next to each name is an ordered list of all the investigator's experiments and the location of that data on the disk. This is analogous to a three-ring notebook to which pages of information are added as required. Additions are made to an ordered index, and the main file need not be ordered.

Inverted list. This uses multiple indexes. Separate files are used for the investigator names and project numbers. These are associated with lists of sample numbers and disk locations where analytical results may be found. A good example of this technique is a thesaurus. This method retrieves information very rapidly, but updates slowly.

Tree. This is a hierarchical approach of storing the data in the database. With this method, it is easy to add things but the search times become long as the number of levels in the tree grow.

Plex. This is a complex tree method. This is analogous to a genealogical tree in which both the mother and father are considered, and in which intermarriage/divorce and re-marriage occurs. Plex structures require storing the data in quite a different form, often invoking a complex set of internal points to thread the data.

Threaded-List Data Storage. This is when entry on a subject points to other volumes and subjects that have related data. To find all of the data associated with a given subject requires beginning at one end and following the thread. Threads update quickly but retrieve slowly.

Hashing. This consists of converting key words such as investigator names, project numbers, etc. to numbers. Then, all alphanumeric expressions that hash to the same number are stored in a separate area along with the location on the disk containing the relevant information. Hashing is used to speed up access to information.

A well-designed LDMS will use a combination of various data storage methods such as inverted files in conjunction with multiple threaded lists, a tree structed ISAM approach, and a hashed access to a multikeyed tree structure. Also, the LDMS would backup (record an entire copy) the database so that the system can be restored if failures occur. In addition, the LDMS should have word processing (text preparation) and list processing (sort, merge, and select) capabilities to aid in preparing reports. Thus, the combination LAN/LDMS will permit efficient resource sharing in the research laboratory and will remove the chemist from the role of transcribing information from one place to another.

III. DESIGN AND IMPLEMENTATION OF A LABORATORY DATA SYSTEM

A. Introduction

The following requirements were considered in implementing the Laboratory Data System (LDS):

-The system should be as inexpensive as possible.

-The system should take advantage of current equipment.

-The system should require a minimum of custom designed hardware.

-The system should be capable of a considerable amount of "number crunching" power for data analysis.

-No single computational task should dominate the system.

-The system should have some graphics capability.

-The system should be flexible and capable of expansion to accomodate new instruments and uses.

-The system should be modular and thus capable of being enhanced at a relatively low cost.

-The system should have some data management system capabilities.

-The system should be capable of supporting a large amount of software development without adversely affecting other functions.

The LDS design that was chosen which met the above requirements was a distributed local area network of microcomputers. This local area network is based on the Corvus Constellation and OMNINET hardware and software designed by Corvus Systems, Inc.

B. LAN Hardware

Corvus Constellation. The Corvus Constellation is a disk based back-end star network[11]. The center of the network is a Corvus Winchester Disk Drive with a 20 Mbytes storage capacity and an average disk access time of 50 msec. The disk has an integral Z-80 microprocessor and thus has a certain amount of intelligence. The Constellation is a multiplexer which connects to the disk drive. Up to eight computers can be connected to the multiplexer, via a ribbon cable of up to fifty feet in length, and all share the same disk drive.

The disk Z-80 microprocessor continuously polls the attached computers to determine if they require service. The Constellation allows either a mix of different microcomputers (excluding LSI-11's) or only LSI-11 computers. Our implementation has two LSI-11 systems and one LSI-11/23 system attached to the Constellation.

Corvus OMNINET. The Corvus OMNINET local network is a distributed control network[12]. Its topology is a semi-bus configuration. It is actually not a true bus configuration because if it were, the failure of any node would not affect the rest of the network. While any of the computers can fail and not affect the network, if the central disk fails, it brings the entire network down. The computers can continue to function using their local mass storage capabilities, but they cannot communicate with the central disk.

Each device attached to the OMNINET has an interface card called a transporter. The transporter utilizes a Motorola 6801 microprocessor which allows it to perform many of the high level network tasks such as the following: message transmission and acknowledgement, error detection and retransmission, and detection of duplicate messages. A collision avoidance scheme is implemented in the transporter to allow any device to tramsmit when the network is available. Collision avoidance is performed using a combination of two methods: the CSMA mechanism is utilized to determine when the network is available and the transporter computes a randomized transmit start time to minimize the probability of two devices trying to access the network at the same time. Since a collision detection mechanism is not required, the OMNINET is implemented on a RS-422 twisted pair wire thus eliminating the cost associated with collision detection hardware.

The OMNINET uses the same 20 Mbyte disk drive as the Constellation. It is connected to the disk drive by an OMNINET disk server which connects to one of the Constellation slots. This disk server is connected to the RS-422 twisted pair cable. The data transfer rate for the Omininet is 128 Kbytes/second and the total allowed length of the cable is 4000 ft. Up to 64 microcomputers can be connected to the network.

Computers in the LAN. There are four APPLE II microcomputers interfaced to the LAN via the OMNINET transporter cards. Each of the APPLE II's has its own local mass storage (at least one mini-floppy disk drive). A Tektronix 4662 Digital Plotter is interfaced to one of the APPLE II's in the LAN. Another APPLE II has a graphics tablet connected to it. A third APPLE II is interfaced to an Hitachi UV-VIS spectrophotometer via an Interactive Strutures 12-bit ADC and a Mountain-Hardware RTC. The fourth APPLE II, with two mini-floppy disk drives, is set up for word processing (Wordstar-from MicroPro, Inc.), record management, and budget calculations. Also, two LSI-11 microcomputers are attached to the LAN. One of the LSI-11 is connected to the LAN via a Constellation Q-bus interface board and the other by the OMNINET transporter card. The LSI-11/23 Interface System is also connected to the LAN with a Constellation Q-bus interface board.

C. System Software

Operating System. Each computer in the system has the capability of using its own operating system. The APPLE II microcomputers use the UCSD P-system for Pascal and Fortran and Dos 3.3 for Basic. The LSI-11 microcomputers are running under the RT-11 operating system.

Graphics. The Plot 10 software package from Tektronix, Inc. is implemented on the LSI-11s. The APPLE II computers use a standardized graphics package called Core_Graph (which was written and implemented on the APPLE II microcomputers by G. Scott Owen using the UCSD P-system Turtlegraphics commands).

Intercomputer-communication. Each computer in the network has the capability of communicating with any other computer or peripheral in the network by a method called Pipes. A Pipe is a buffer on the central disk that collects data from a sender and makes it available to a receiver. Senders and receivers may be different programs on different computers running at different times. The only restriction is that the sending computer must send all data before the data is available to the receiving computer.

In addition to the Pipes, computers attached to the LAN with the OMNINET transporter can send messages to any device in the OMNINET network or broadcast messages to all devices attached to a transporter in the network. The transporter accepts two major catagories of commands from the host microcomputer to control the flow of messages: the send message command which transmits a message up to 2047 bytes in length to any designated host socket; and the setup receive command which prepares the host sockets to receive incoming messages.

Data Acquisition Programs. The data acquisition programs are routines that acquire information from the different chemical instruments and perform different digital filtering techniques. Some filtering programs are boxcar averaging, ensemble averaging, Savitzky-Golay weighted filter, and Fast Fourier Transform filter.

Data Analysis Programs. Some of the data analysis programs are: curve deconvolution program using the Taylor series method and the FFT method, multicomponent analysis program, a program to determine the secondary structure of a protein from the CD spectrum, and a program to calculate and plot the molar ellipticities for CD spectra.

D. Discussion

The design that was chosen for the implementation of the LDS was a distributed local area network of microcomputers with the chemical research instruments interfaced to the microcomputers. The network has a powerful intercomputer communication protocol which allows data collected from one microcomputer to be sent to another more powerful microcomputer system for analysis. If necessary, the data can be sent to a minicomputer or even a super minicomputer. The local area network design is based on the Corvus Constellation and OMNINET schemes.

The Corvus Constellation was available about a year before Corvus Systems Inc. announced the OMNINET network. Thus, the Constellation system was purchased and implemented with two LSI-11 systems and one LSI-11/23 system. The LSI systems are connected to the Constellation multiplexer via Corvus Systems Q-bus interface boards. The Q-bus interface board plugs into the Q-bus backplane slot of the LSI-11 computers. A 34 pin flat cable connects the interface board to the multiplexer. The multiplexer is connected to the 20 Mbyte Disk Drive by another 34 pin flat cable.

The 20 Mbyte disk was configured to appear to be four logical RL01 disk drives (5 Mbytes each) to the LSI-11 systems. The first logical disk drive (0) was the system disk for the LSI-11/23. The second logical disk drive (1) was the system disk for the two LSI-11s. This was necessary because, even though the LSI-11 and 11/23 have the same basic instruction set they have different floating point instruction sets. The LSI-11 has the EIS/FIS instruction sets and the LSI-11/23 has the FPU instruction set. Thus, the LSI-11 and 11/23 can use the same system programs and utilities but they require different compilers and the associated libraries (for Basic, Fortran, and Pascal). The compilers and libraries for the EIS/FIS instruction set were placed on logical drive 1 and those for LSI-11/23 were placed on logical drive 0.

Four spectrophotometers were interfaced to the LSI-11/23 (details of this are given in the next chapter). The two LSI-11's are used for program development and data analysis. Although the Constellation scheme met the requirement for a successful LDS, it has two drawbacks. First, the attached computers must be within 50 feet of the multiplexer. This is a major problem since all of the instruments are not within 50 feet of each other. Second, as previously stated, the Constellation can be attached to a variety of microcomputers (APPLE II, PET, TRS-80, S-100 BUS, etc.) or to LSI-11 systems. That is, one can not mix LSI-11 systems and other computers with the Constellation.

Later, Corvus Systems Inc. announced the OMNINET system. This seemed to be the next approach to take. The Corvus was purchased with the understanding that it could be connected to the existing Constellation network. But this wasn't apparent when the OMNINET was first received since the existing software for the Constellation was incompatible with the OMNINET. With much effort on our part plus repeated calls to the company, Corvus Systems finally sent the software that allowed the OMNINET to be used with the Constellation. It was a considerable task to set up the network because no one else had ever mixed a Constellation system with an OMNINET system.

The OMNINET is connected to the 20 Mbyte Disk Drive by a Disk Server which connects to one of the Constellation multiplexer slots via a 34 pin flat cable. This Disk Server is connected to the OMNINET bus which is a twisted pair cable that sends/receives data in the RS-422 protocol. The OMNINET uses the CSMA scheme to control access to the network. The data transfer rate for the OMNINET is 128 Kbytes/second and the total allowed length of the cable is 4000 feet. Up to 64 computers can be connected to the network via special interface cards called transporters. At this time, transporter cards are available for APPLE II, IBM PC, and LSI-11 microcomputers. Furthermore, Corvus has no available software for the LSI-11 transporter. Software which allows the LSI -11 transporter to send messages via the OMNINET bus was written in our laboratory (in MACRO - 11).

IV. INTERFACING SEVERAL SPECTROPHOTOMETERS TO AN LSI-11 COMPUTER

A. Introduction

An attempt was made to interface four spectrophotometers to an LSI-11 microcomputer. The instruments were a Varian Model 3700 Gas Chromatograph, a Cary 17 UV-VIS-NIR Spectrophotometer,a Beckman 4240 Infrared (IR) Spectrophotometer, and a Durrum-Jasco J-20 Circular Dichroism (CD) Spectrophotometer. All of these instruments output an analog signal. In order to interface these intruments to the computer, the analog signal has to be digitized before it can be stored into computer memory. This process of digitizing the analog signal is performed by an analog to digital converter (ADC).

The ADC is an electronic circuit that converts an analog input into a n-bit binary number. The basic conversion scheme for a typical succesive approximation ADC, such as we used, is shown in Figure 11. An unknown input voltage V_X is connected to one input of an analog signal comparator, and a time dependent reference voltage V_R is connected to the other input of the comparator[13]. V_R is varied until V_X is determined and the comparator determines whether the output voltage corresponds to a logic 1 or a logic 0.

The number of bits of the ADC determines its resolution. For instance, a 12-bit ADC has a resolution of 1 part in 4096 times the fullscale voltage e.g. a 12-bit ADC with a 10v fullscale range has a resolution of 2.44mv. ADCs are available with resolutions varying from 6 to 20 bits.

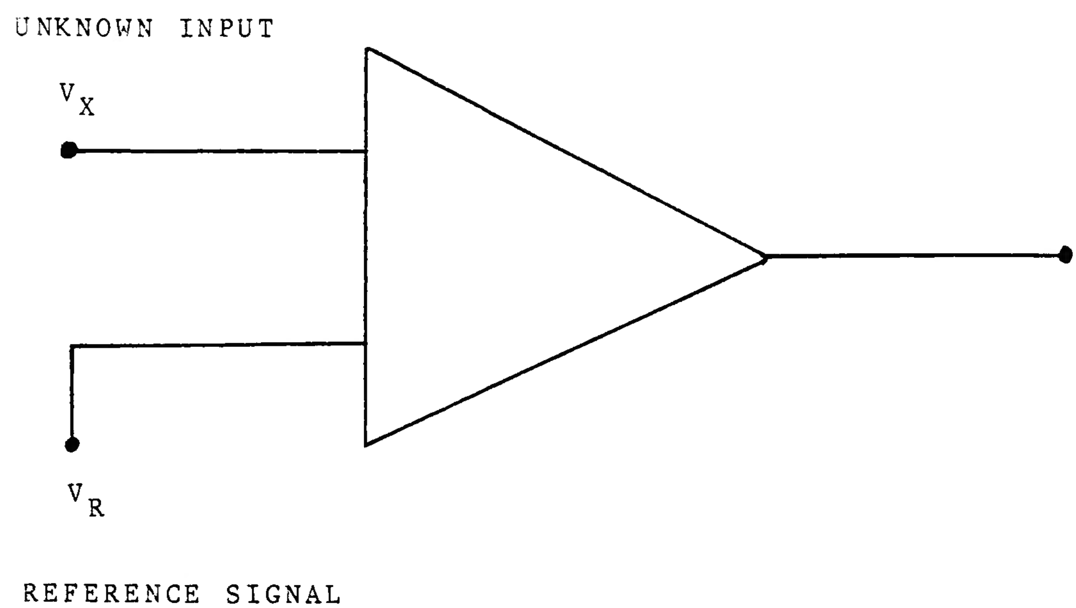


Figure 11. The basic conversion scheme for a typical ADC.

B. Materials

Analog Input/Output System. The analog I/O system was purchased from Data Translation Inc. (Model DT2781). This is a DEC Q-bus dual height card that contains a 16 channel multiplexer (8 differential channels), an instrumentation amplifier to provide a high input impedance and noise rejection, a sample-and-hold that acquires and maintains a fixed output corresponding to the input level, two channel DAC (digital to analog converter), and an ADC.

The ADC has two input ranges: 0v to 10v (unipolar) fullscale with 12 bits resolution and -10v to +10v (bipolar) fullscale with 11 bits resolution. The conversion time is 25 microseconds. Also, the DT2781 has two external trigger sources for the synchronization of the A/D conversion: RTC INL (pin 21) and EXT TRIG (pin 19). Each of the two sources can be enabled or disabled individually by software by setting or clearing certains bits of the control and status register.

Real Time Clock. The Real Time Clock is also from Data Translation Inc. (Model DT2769). The DT2769 has a base frequency of 10 MHz which is divided into five rates (1 MHz, 100 KHz, 10 KHz, 1 KHz, and 100 Hz). It is used to generate an A/D (analog to digital) conversion at a desired frequency.

Gas Chromatograph. A modular, dual - column Varian Model 3700 Chromatograph is used for gas chromatography measurements. On the instrument input/output panel (located on the rear panel), there are two 0v to 10v computer outputs. A connection was made from one of the output connectors (upper det mode) to one of the input channels on the analog I/O card in the LSI-11 computer.

Infrared Absorbtion Spectroscopy. A Beckman 4240 Infrared (IR) Spectrophotometer is used for infrared absorbtion measurement. This instrument has a wavenumber range of $4000cm^{-1}$ to $250cm^{-1}$ with eight scanning speeds (1000, 600, 300, 150, 50, 20, 5, and $2cm^{-1}/min$).

Located on the side of the instrument is a terminal block (TB201) used for connection of certain accessories and for analog and digital output. The computer connections were made to the pins marked %T (per cent transmittance), ABS (infrared absorption), and SIG GND (signal ground) of the terminal block.

Circular Dichroism Absorbtion Spectroscopy. A Durrum-Jasco J-20 Circular Dichroism (CD) Spectrophotometer is used for the CD measurement. The CD spectrophotometer is a null system utilizing gain modulation to continuously balance and record the CD signals. High level illumination is provided throughout the spectral range (800nm to 185nm) by a 450 watt high pressure Xenon arc lamp.

A previously designed special interface which was used to connect the CD instrument to a Nicolet Signal Averager was used to interface the spectrophotometer to the computer. This special circuit, housed in a black box, has two BNC outputs. The BNC labeled "S" on the black box is for the CD signal while the other connector is for a trigger signal. The trigger signal goes from 0v to 15v when the instrument is placed in the scan mode. This signal is used by the CD program to trigger the start of the A/D conversion.

UV-Visible Absorbtion Spectroscopy. A Cary 17 UV-VIS-NIR Spectrophotometer is used for the UV-Visible absorbtion measurements. The Cary 17 has a wavelength range of 186nm to 2650nm. It is equipped with a Universal Absorbance- %T Slidewire which gives eight absorbance ranges and a 0 to 100% T range.

A special circuit designed by Mr. Armen S. Karapetain, electrical engineer with Lockheed-Georgia, was used to help interface the Cary 17 to the computer. This circuit was connected between the Cary 17 photometric interface connector J53B and two channels on the ADC. One channel was used for the sample signal while the other was for the reference signal. Another circuit was designed and connected between the J53B MVST pin and the DT2781 EXT TRIG pin to provide timing for the ADC at sample time.

Computer System. The computer system was purchased from

Netcom Products, Inc. This includes a Netcom's HV-1123 enclosure assembly, a KDF11-AA microprocessor, and memory modules. The HV-1123 includes a Q (LSI-11) bus backplane, power supplies for the processor, memories and peripheral devices. The KDF11-AA (LSI 11/23) is a 16-bit high-performance microprocessor contained on one dual-height multilayer module. The KDF11-AA has an FP11 hardware floating point unit to speed up arithmetic calculations, an FT11 memory management unit for 256 Kbytes of protected multi-user program space, and an extended instruction set (EIS). The memory is contained on a dual-height multilayer card and the card has 32K, 18 bits of RAM(random access memory). There are four memory cards in the system for a total of 256 kbytes (128K words) of RAM.

The system configured is a LSI-11/23 with 128K words of RAM. An ADM 3A (with a Retrographics Tektronix compatible graphics board with a 512 x 256 pixel resolution) is used as the console terminal. Also, the LSI-11/23 computer has a Heathkit dual floppy disk system (RX01 compatible 256 Kbytes/drive) and a Heathkit Data Systems printer. This system is connected to the LAN and uses the CORVUS Winchester Disk Drive (as two RL01: drives).

<u>Other Perpherals</u>. A Tektronix 5110 Oscilloscope is used for real time graphics output. For a hard copy of the graphics, a Houston Instrument Omnigraphic X-Y Plotter is connected to

the system. Both of these instruments were interfaced to the two DAC channels.

DTLIB Real-Time Peripheral Support. DTLIB is a software package obtained from Data Translation, Inc. It consists of a single library of subroutines that operates under the RT-11 operating system. DTLIB initiates and controls the function of the DT2781 and DT2769 boards.

C. Discussion

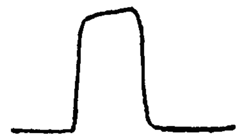
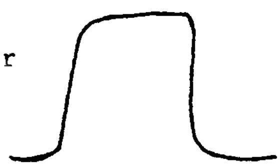
With the assistance of Mr. Armen S. Karapetain, electrical engineer with Lockheed-Georgia Company, it wasn't difficult to connect the Varian Model 3700 Gas Chromatograph and the Beckman 4240 IR Spectrophometer to the LSI-11/23 computer system since both of these instruments have a continous analog signal at their computer interface connectors. However, the Cary 17 and the Durrum-Jasco J-20 spectrophotometers have a pulsating analog signal at their computer interface connectors. This presented a problem when trying to synchronize the scan rate of the instrument with the timing of the RTC to signal the DT2781 to perform an A/D conversion.

For the CD spectrophotometer, a previously designed interface circuit which was used to connect the instrument to a Nicolet Signal Averager[14] was used to connect the computer to the instrument. This circuit was connected to the CD recorder amplifier output. Since the CD scale potentiometer controls the recorder amplifier output, the CD program has to compensate for the different voltage output by dividing the digitized voltage by the scaling factor in order to obtain the correct CD measurements. Also, this circuit has an output voltage of 0v to 15v while the DT2781 analog board has a maximum input voltage of 10v. This presents a problem only when the CD scale is set to a factor of 10 or greater. To compensate, a scale factor less than 10 should be used or another circuit could be designed to match the CD interface circuit output voltage to the DT2781 input voltage.

The output signals for the Cary 17 photometric interface connector J53B are shown in Table II. This is a 6 pin receptacle and the signals available are: pin A is the sample out; pin B is ground for both the sample and reference signals; pin C is the reference out; pin D is a 30 Hz multivibrator (MVST) timing signal which is more positive at sample time; pin E is a 30 Hz multivibrator (MVRT) timing signal which is more postive at reference time; and pin F (MVGND) is ground for MVST and MVRT signals. The maximum height of the sample pulse is 6v and the reference pulse is 5v when there is no absorbing species in the sample or reference compartment. The period of the sample and reference pulses is 32 ms while the pulse width is 8 ms and there are 16 ms between the sample and reference pulses. Furthermore, when an absorbing compound is placed in the sample cell compartment, the height of the sample pulse will decrease and the amount of decrease is used to indicate the amount of absorption by the sample.

Pins MVST and MVRT are used to provide timing for the sample and reference pulses, respectively. The MVST pin is

Table II. Cary 17 Photometric Interface Connector J53B

Pin	Signal	Discription	Waveform[a]
A	SAMPLE	Sample amplifier ac feedback point.	
B	SIG. GND	Ground return and shield for SAMPLE and REF signals.	
C	REF	Reference amplifier output.	
D	MVST	30 Hz multivibrator more positive at sample time.	
E	MVRT	30 Hz multivibrator more positive at reference time.	
F	MVGND	Ground return and shield for MVST and MVRT signals.	

[a]Maximum peak height for SAMPLE = 6v; REF = 5v; MVST and MVRT = 19v.

connected to the EXT TRIG of the DT2781. The sample pulse occurs when the MVST signal goes from a low voltage to a high voltage (or positive transition). This is the time to signal the DT2781 to do an A/D conversion. However, the two trigger inputs of the DT2781 are only edge sensitive and initiate a conversion on the negative (high to low) transition of the input signal. When the negative edge of the MVST signal occurs, the sample pulse has already passed as depicted in Figure 12. If an A/D conversion is performed at this time an improper reading will be obtained. In order to correct this problem, a special circuit was designed. The schematic of this circuit is shown in Figure 13. The circuit is called a one-shot monostable multivibrator because it produces one output pulse for each input pulse. One application of the monostable multivibrator is to produce a pulse of some specific duration. Since this circuit produces one output pulse for each negative going input pulse, the output frequency is the same as the input frequency. The pulse width of the output is determined by the R_AC_1 time (R_A-resistor A and C_1-capacitor 1) constant. The pulse width is approximately:

$$PW = 1.1 \times R_AC_1.$$

The MVST signal is connected to the input of the monostable multivibrator circuit. The negative going pulse of

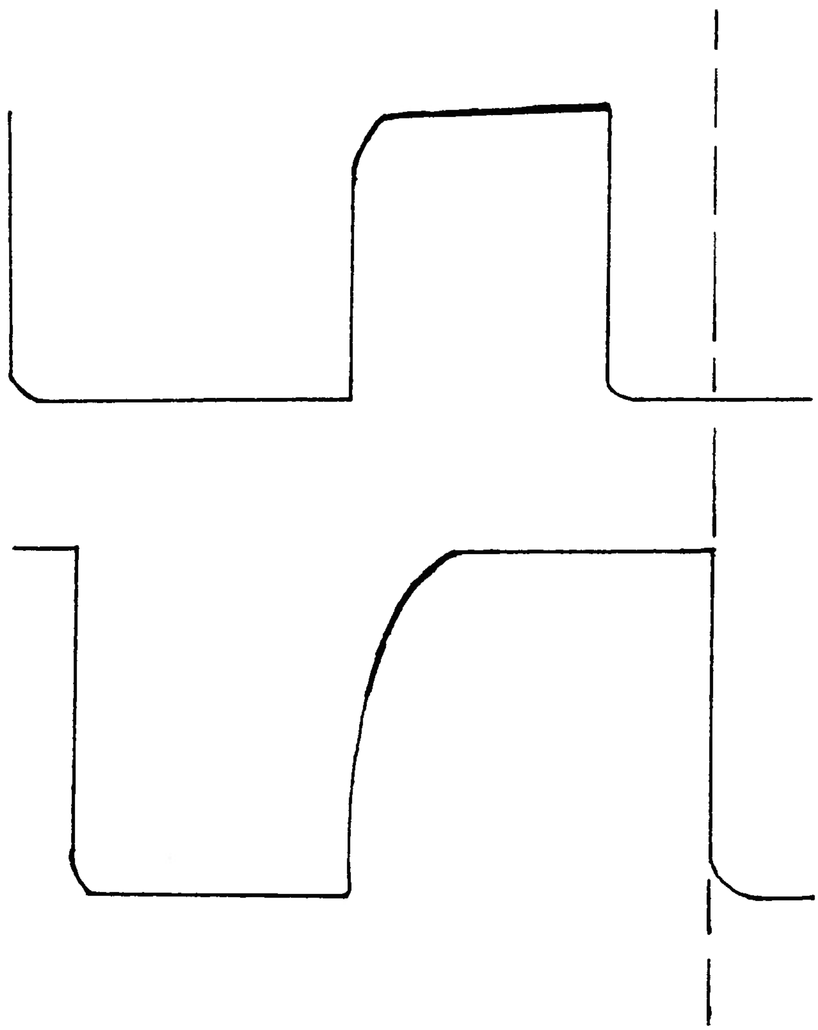
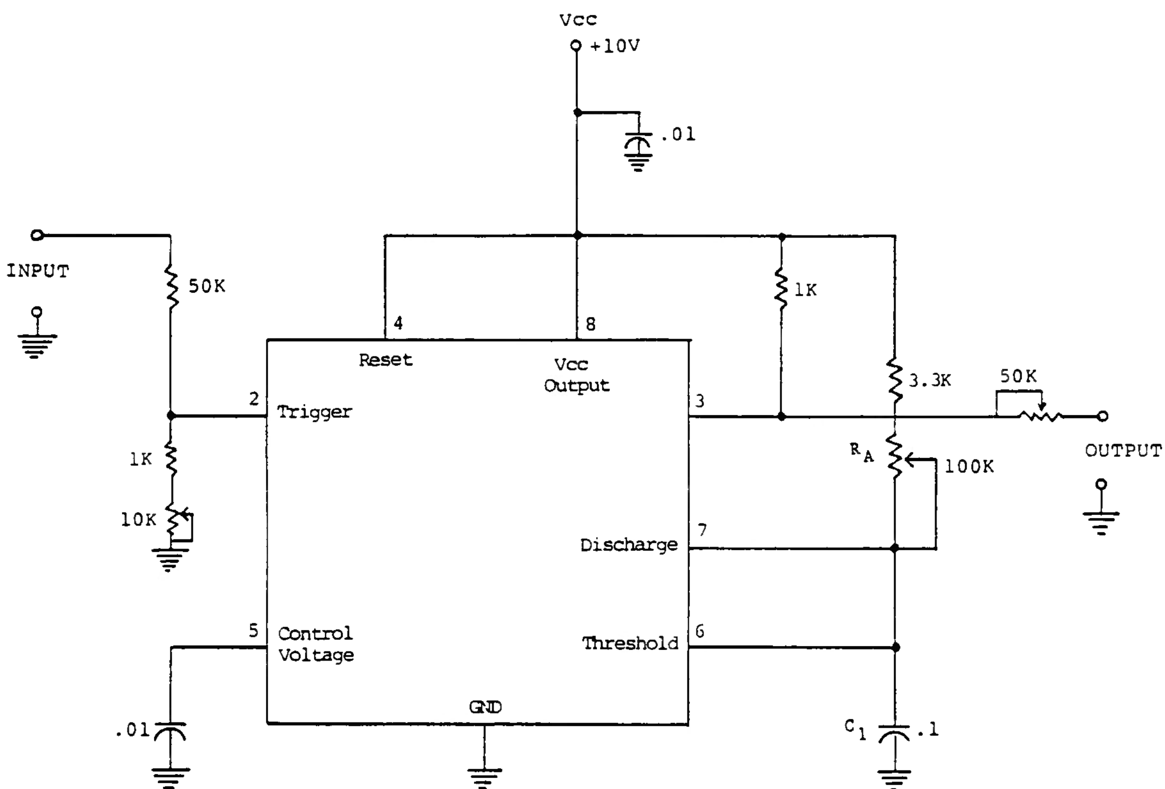


Figure 12. The negative (high to low) transition of the MVST signal doesnot correspond to the sample output.



<u>Figure 13.</u> A one-shot monostable multivibrator circuit.

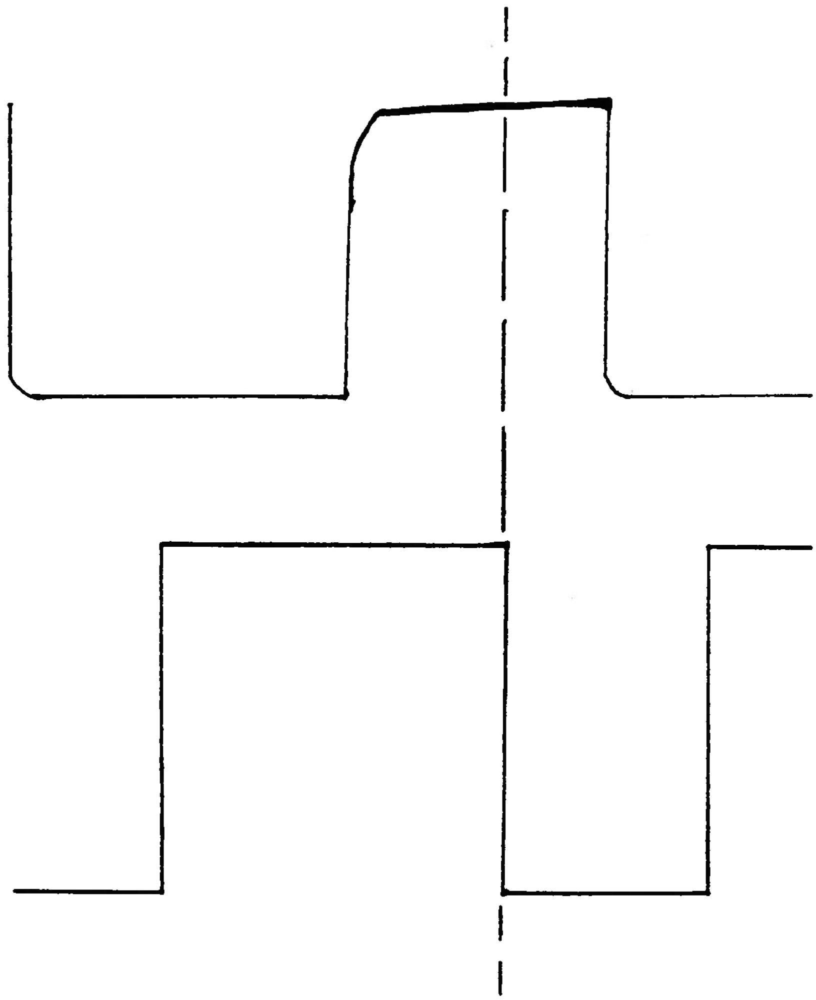


Figure 14. The output pulse of the monostable multivibrator adjusted so that the negative end of the pulse corresponds to the sample pulse.

the MVST signal triggers the circuit to produce a positive output pulse. The output pulse width can now be adjusted using a variable resistor at R_A to change the $R_A C_1$ time constant so that the negative end of the pulse corresponds to sample time. The output of the monostable circuit is then connected to the EXT TRIG of the DT2781 board. The negative end of this pulse would trigger the A/D board (if enabled by software by setting bit 4) to take a conversion at sample time. This is shown in Figure 14. Instead of designing another monostable multivibrator circuit for the MVRT timing of the reference signal, the timing of the reference pulse was performed with the RTC. Since the reference pulse occurs 16 ms after the sample pulse, the RTC was programmed to signal the A/D board to do a conversion 16 ms after the DT2781 was triggered by the monostable multivibrator circuit.

The subroutine UVCON which acquires the data points from the Cary 17 and controls the functions of the DT2781 and DT2769 boards is written in MACRO assembly language (developed under the RT-11 operating system -see APPENDIX for complete listings). First, the RTC is programmed to run in a single interval mode (Mode 0). This is done by loading the Buffer-Preset Register (RTCBPR) with the 2's complement of the required number of clock pulses to be counted that will generate the amount of time delay (16ms) required at a selected clock frequency. Loading the RTCBPR with -16 at a clock frequency of 1 KHz would generate a 16 ms time delay.

```
        MOV   #-16,RTCBPR     ;Load the Buffer/Preset Register
                              ;with the desired number of clock
                              ;pulses.
        MOV   #140,RTCCSR     ;Set MODE bits = 0, Set IRATE bits
                              ;= 1KHz, enable clock overflow bit
                              ;(bit 7) to generate an interrupt.
        MOV   #OVFINT,CLKVEC  ;Move overflow interrupt handler
                              ;into the clock vector address.
```

Next, the DT2781 Control - Status Register (ADCSR) is programmed to start a conversion on channel 6 from the monostable multivibrator external pulse circuit.

```
        MOV   #ADINT,ADVEC    ;Move A/D Done interrupt handle
                              ;into DT2781 vector address
        MOV   #6,STCH         ;Channel 6
        SWAB  STCH            ;Put channel no. into high byte.
        BIC   #377,STCH       ;Clear low byte.
        ADD   #120,STCH       ;Add gain bits = 0, EXT TRIG ENB
                              ;bit = 1 (set), and DONE INT ENB
                              ;bit = 1 (set) to low byte.
        ;
        ;
        MOV   STCH,ADCSR      ;Move parameters into Control-
                              ;Status Register --- Wait for
                              ;Ext Trig to start a conversion on
                              ;channel 6.
        ;
REPEAT: CALL  DISP            ;Display data on the scope.
        CMP   DCOUNT,#0       ;Is the number of point = zero?
        BGT   REPEAT          ;No - do another conversion.
        ;
        ;
RETURN: RTS   PC              ;Yes - return to main program.
;
;    A/D DONE INTERRUPT HANDLER
ADINT:  MOV   ADDBR,(R3)+     ;Move data from Data Buffer
                              ;Register to data buffer.
        INC   RTCCSR          ;Start RTC to count.
        XOR   #400,ADCSR      ;If channel 6 set to 7
                              ;else if channel 7 set to 6!!!!!!
```

```
                         ;Channel 6 - sample signal.
                         ;Channel 7 - reference signal.
        DEC   DCOUNT     ;Decrement the number of points
        RTI              ;Leave A/D interrupt handler.
;
;    RTC OVERFLOW INTERRUPT HANDLER
OVFINT: INCB ADCSR       ;Start A/D conversion on REF SIG!!
        RTI              ;Leave RTC OVF interrupt handler.
;
```

Figure 15 shows a block diagram of the interface system involving the spectrophometers and the LSI-11 computer. The functions of the DT2781 and DT2769 boards can also be controlled by the DTLIB software as described below:.

> SETR. The SETR subroutine is used to set the clock to pulse at a particular frequency. This pulse is then sent to the DT2781 board via pins RR and SS to trigger the A/D conversion.
>
> RTS. The RTS subroutine is used to initiate and control real time sampling of the analog input channel. Collected data are stored in an input buffer (a single dimensional array).

The subroutines that output the data to the X-Y plotter and the oscilloscope via the two DAC channels are written in MACRO-11 assembly language.

> XYPLOT. The XYPLOT routine plots an array of data on the X-Y plotter.
>
> DISP. The DISP routine displays the data on the oscilloscope while the data is being collected from the instrument.

The program that acquires the data from the individual instruments in the system is called AMLAB. The main program

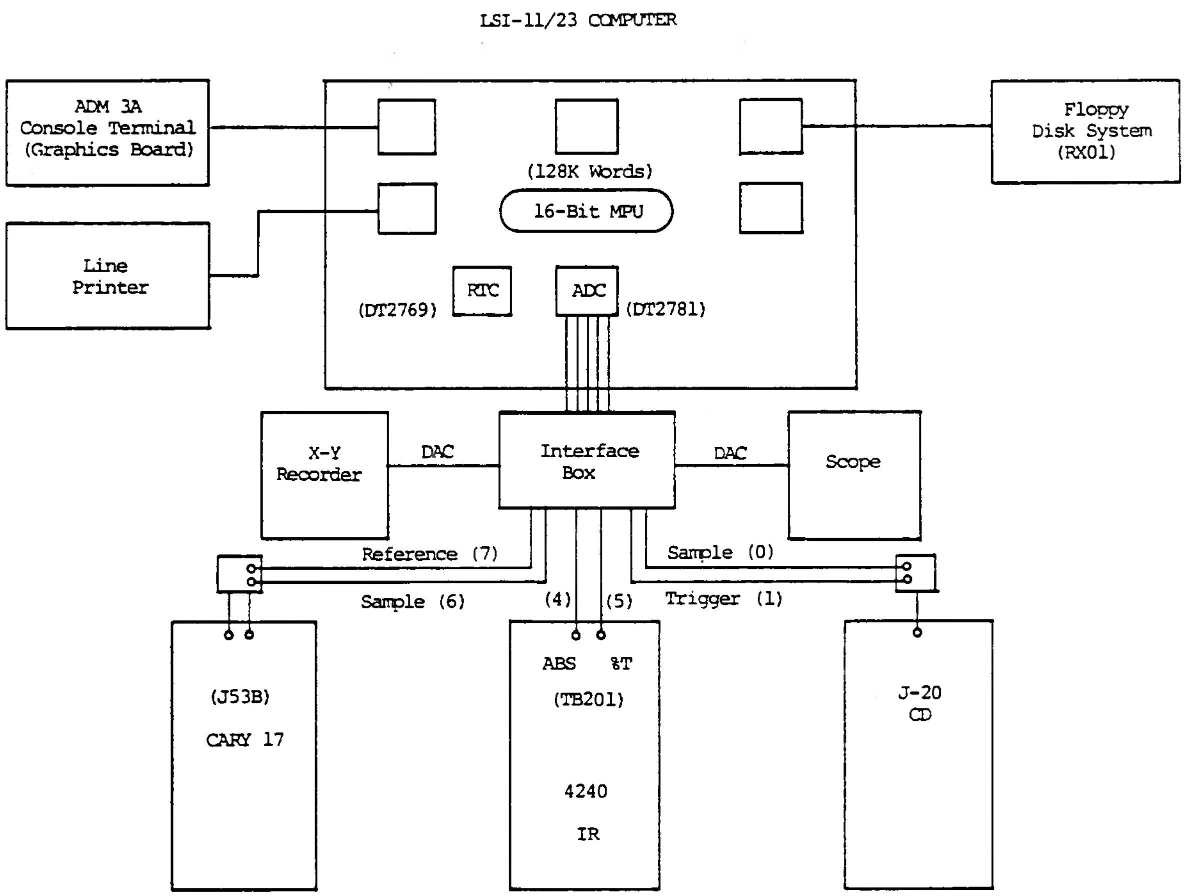


Figure 15. Block diagram of the LSI-11/23 interface system.

of AMLAB is written in FORTRAN with some subroutines in FORTRAN and some in MACRO-11. When the program is excuted, a menu is displayed allowing the user to choose which instrument the computer should acquire the information from. After collecting the data, another menu appears allowing the user to either print the data on the line printer or terminal, store the data on the disk in a file, plot the data on the X-Y plotter, or quit the program. A complete listings of AMLAB is given in the APPENDIX.

V. DISCUSSION

The chemistry research laboratories are primarily housed on the third floor of Charles Merrill hall. The LAN RS432 cable runs from one end of the Hall to the other end as shown in Figure 16 (the dotted line in the figure represents the RS-422 cable). This figure shows the floor plan of the third floor and the location of each laboratory. The head-end of the cable is located in room 302. This room contains several research instruments including the spectrophotometers that are interfaced to the LSI-11/23 system. Also, the 20 Mbyte Winchester Disk Drive connected to the Corvus Constellation is in this room. The terminal-end of the cable is in Dr. Owen's office which is located in room 316.

A block diagram of the LAN with the computers and peripherals connected to it is shown in Figure 17. There are three different types of computer systems in the LAN (APPLE II, IBM PC, and LSI-11 systems) each of which use a different operating system. Therefore, the 20 Mbyte disk was partitioned to accommodate each type of computer and its respective operating system (or systems): 4 1/2 Mbytes for APPLE II U.C.S.D. Pascal, 1/2 Mbytes for APPLE II CP/M, 1 Mbyte for APPLE II DOS 3.3, 3 Mbytes for IBM PC-DOS, and 10 Mbytes for the LSI-11 systems (emulating two DEC RL01 disk

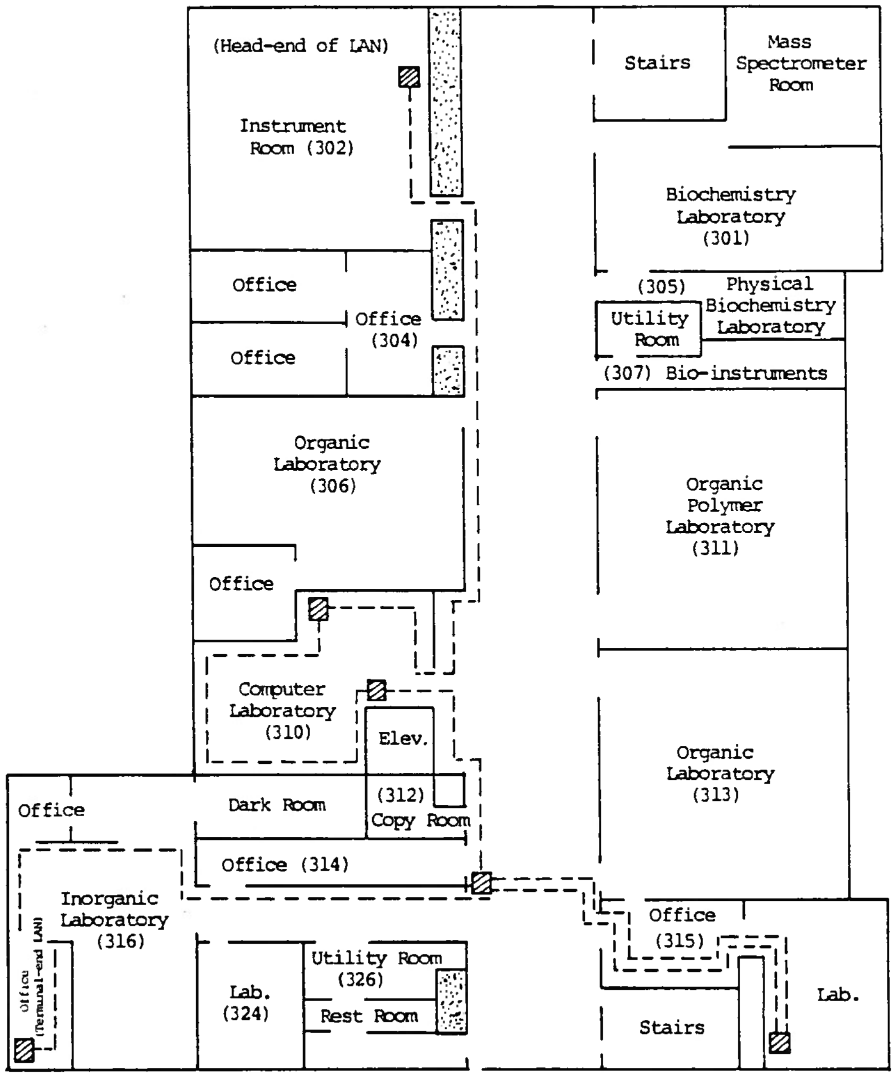


Figure 16. Diagram of the 3rd floor of Charles Merrill Hall.

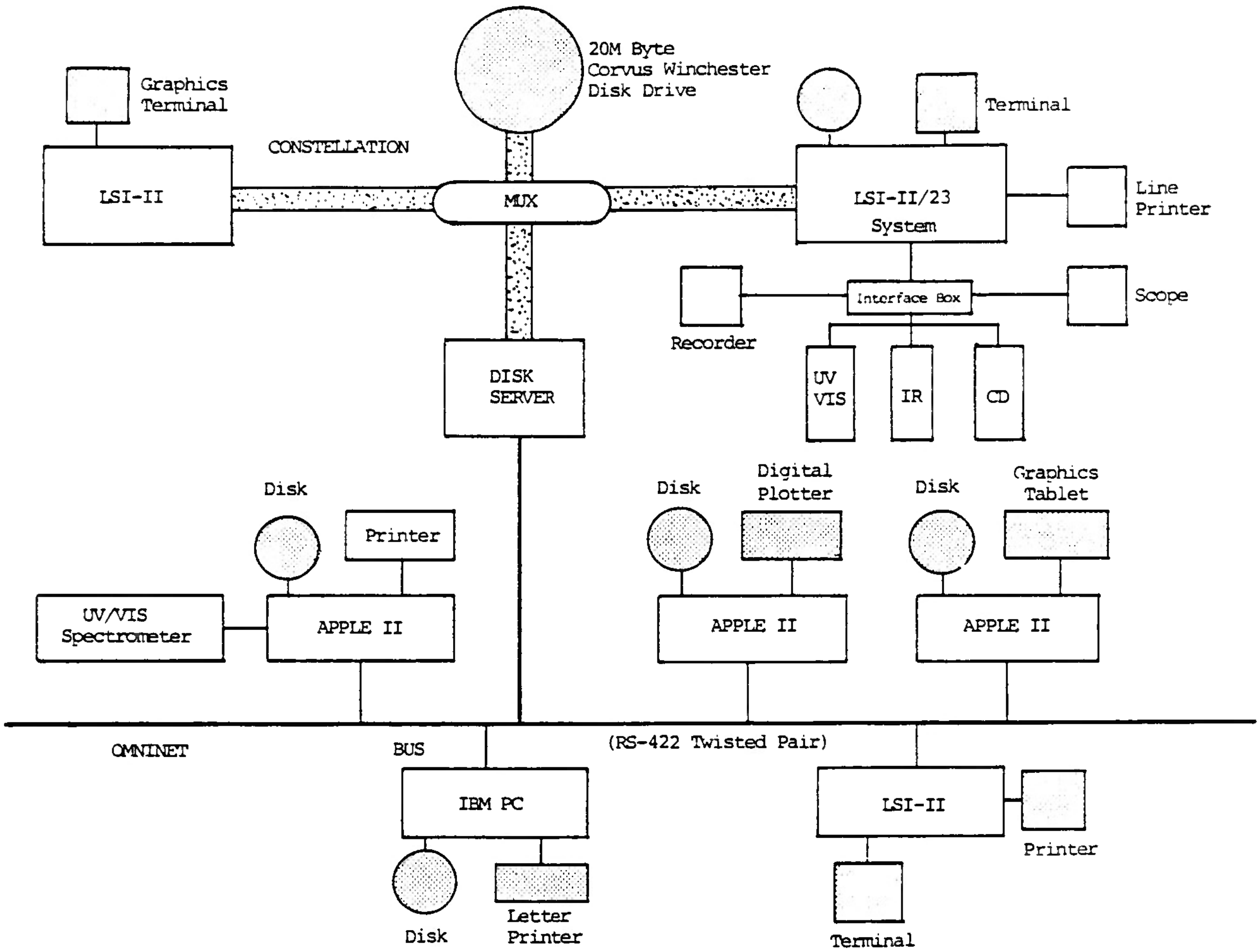


Figure 17. Block diagram of the LAN.

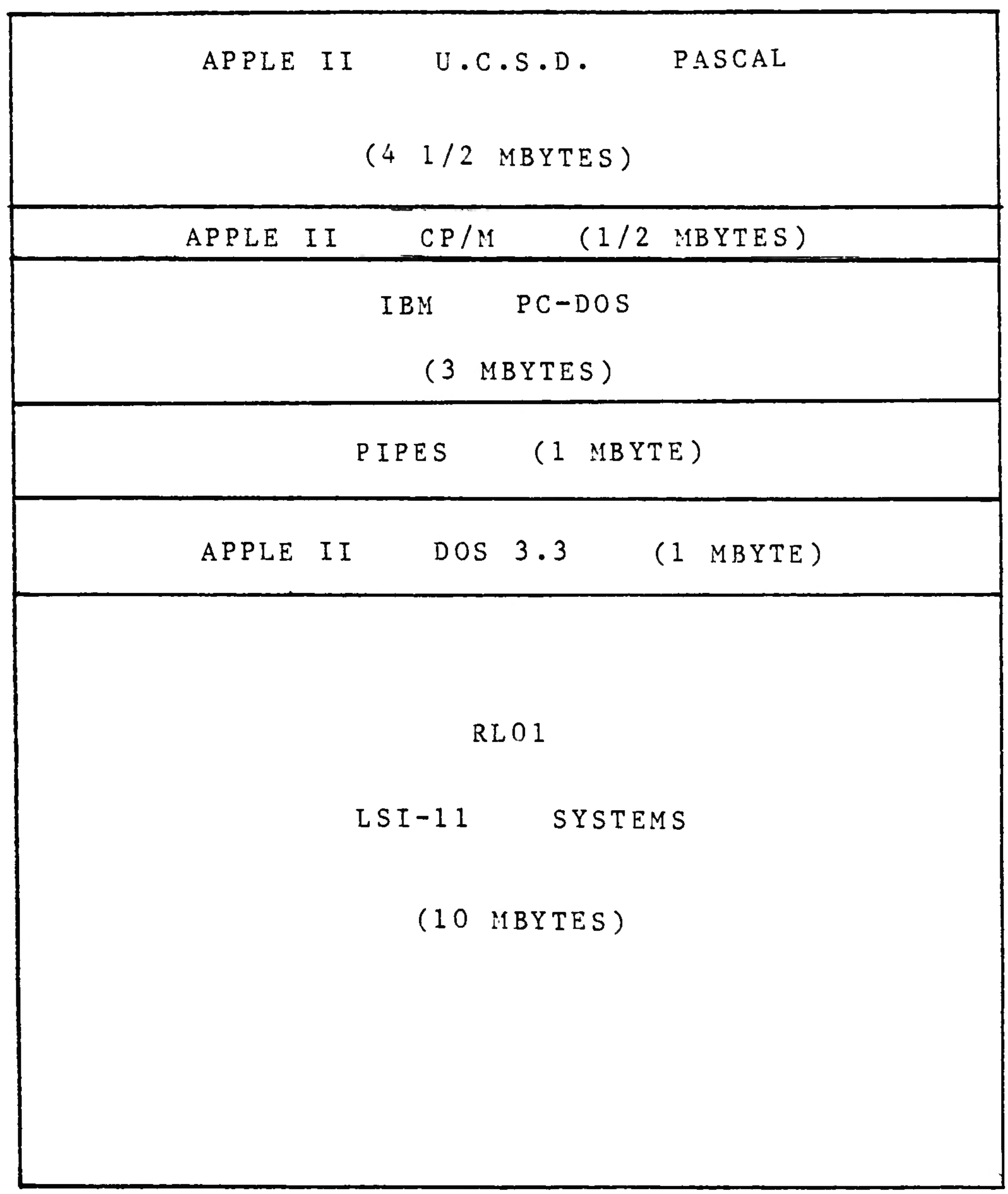


Figure 18. Partition of the 20 mbyte disk.

drives). This is shown in Figure 18. The remaining disk space (1 Mbyte) is used for the PIPE area.

LDS Applications

Entering The System. Each computer system has it own log-on procedure to let the user enter the system. For the APPLE II System, the user must have a 1 to 4 character password (and a 2 character ID) in order to have access to the system.

Data Acquisition. Experimental data are acquired from the LSI-11/23 Interfaced System by the program AMLAB. Because of the continuous failure of either the instruments interfaced to the system or the LSI-11/23 computer, this program could not be tested to determine whether or not it produces adequate results. The programs ADUV8 (for 8-bit data acquisition) and ADUV12 (for 12 - bit data acquisition) written by Dr. G. S. Owen and modified by Dr. James Currie acquire data from the APPLE II interfaced Hitachi UV-VIS spectrophotometer. Each of these programs has the capability of performing real-time digital filtering to enhance the S/N ratio.

The data acquisition programs can store the experimental data in a database on the 20 Mbyte disk. The information is stored on the disk in the following format:

Investigator Name (30 characters max.);

Title of Experiment (30 characters max.);

Date (20 characters max.);

Instrument Type (20 characters max.);

Attributes (nul, raw data, boxcar average, ensemble average, SGsmoothed, FTsmoothed);

X-interval (a real value);

Y-scale (a real value);

Data (an array of integer values).

After the experimental data are stored on the disk, the post-run filter programs can be used to further improve the S/N ratio. The next series of figures illustrate how digital filtering programs can remove noise from the signal. Figure 19a shows a computer simulated spectrum using a double sine angle function. Noise was added to the spectrum by a random function (Figure 19b). The spectrum in Figure 19c is the result after a 16 scans ensemble average and a 25 points Savitzky-Golay filter were performed, thus enhancing the S/N ratio by a factor of 20. Figures 20 to 22 illustrate the digital filtering capability of the FFT filter program. Figure 20 shows a graphical representation of a sine wave function. Figure 21 depicts the sine wave with noise added to it. A forward transformation was performed on the noisy data. While in the frequency domain, a optimal zero filling range is inputted into the real and imaginary array. Figure 22 illustrates the results of the inverse transformation of

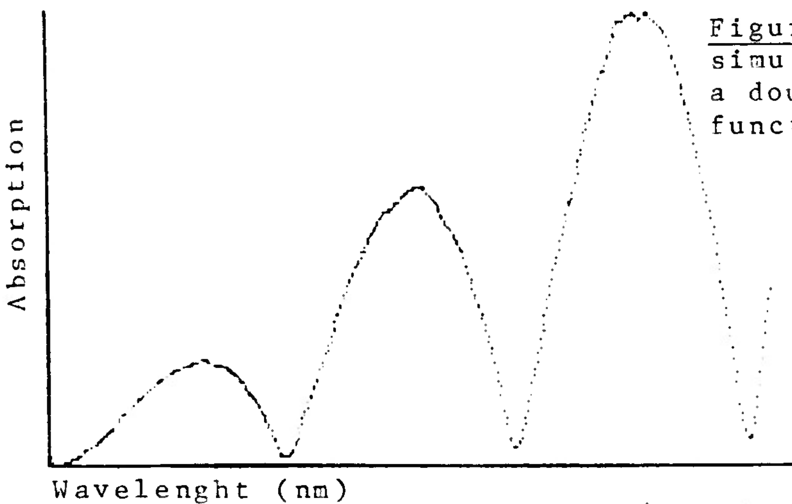


Figure 19a. Computer simulated spectrum using a double sine angle function.

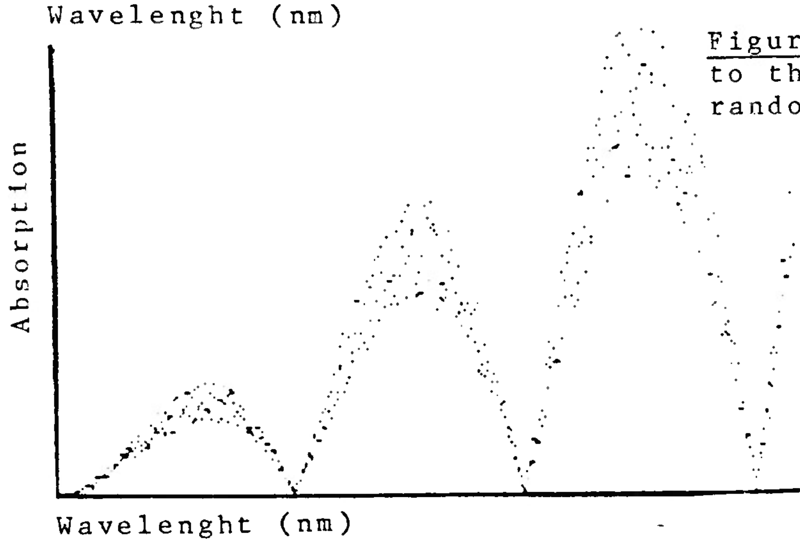


Figure 19b. Noise added to the spectrum by a random function.

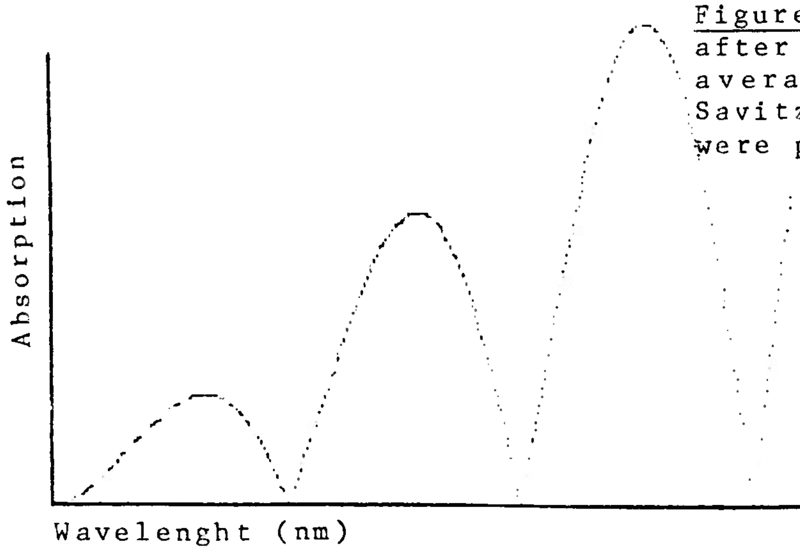


Figure 19c. The result after a 16 scans ensemble average and a 25 points Savitzky-Golay filter were performed.

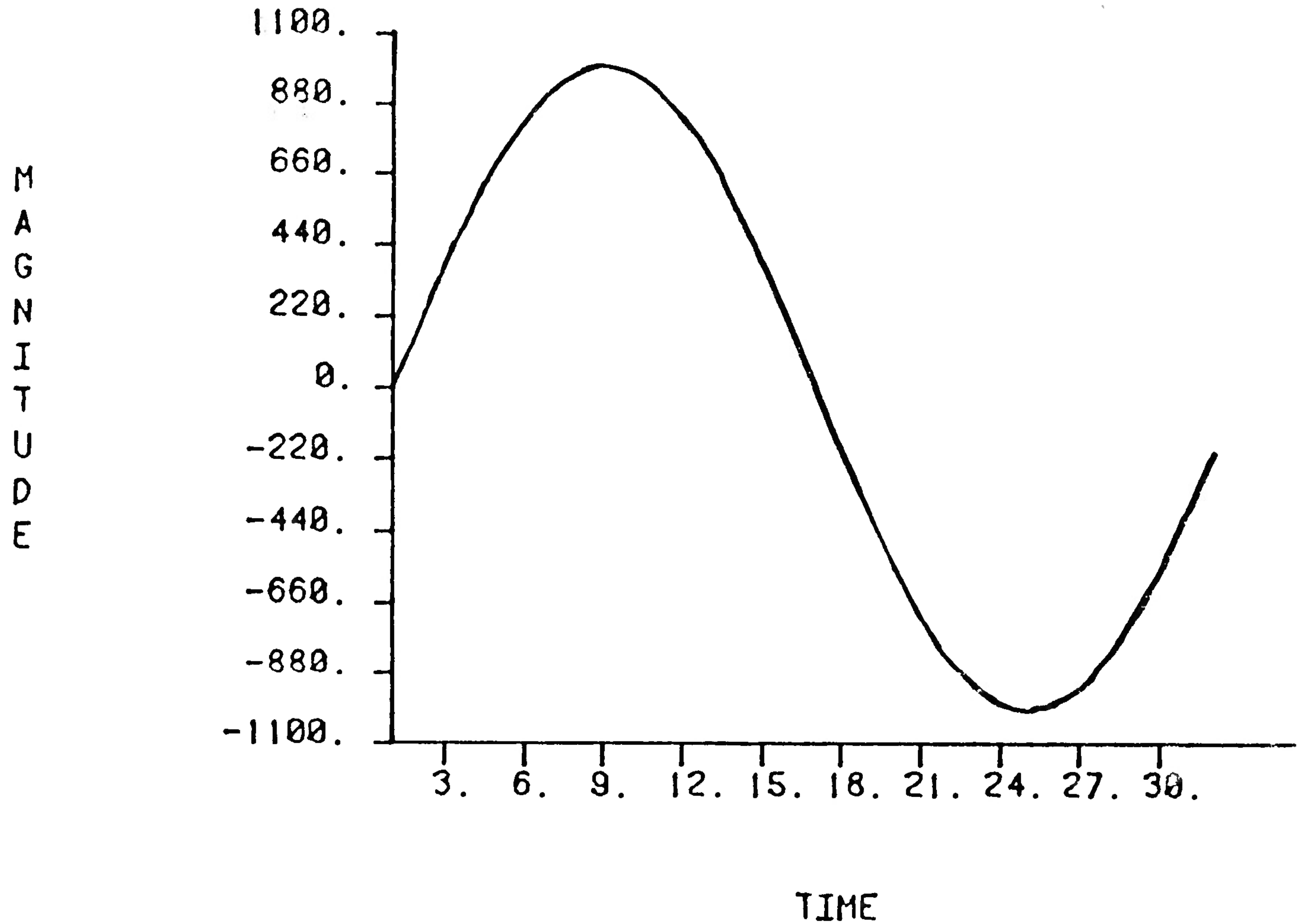


Figure 20. A graphical representation of a sine wave function.

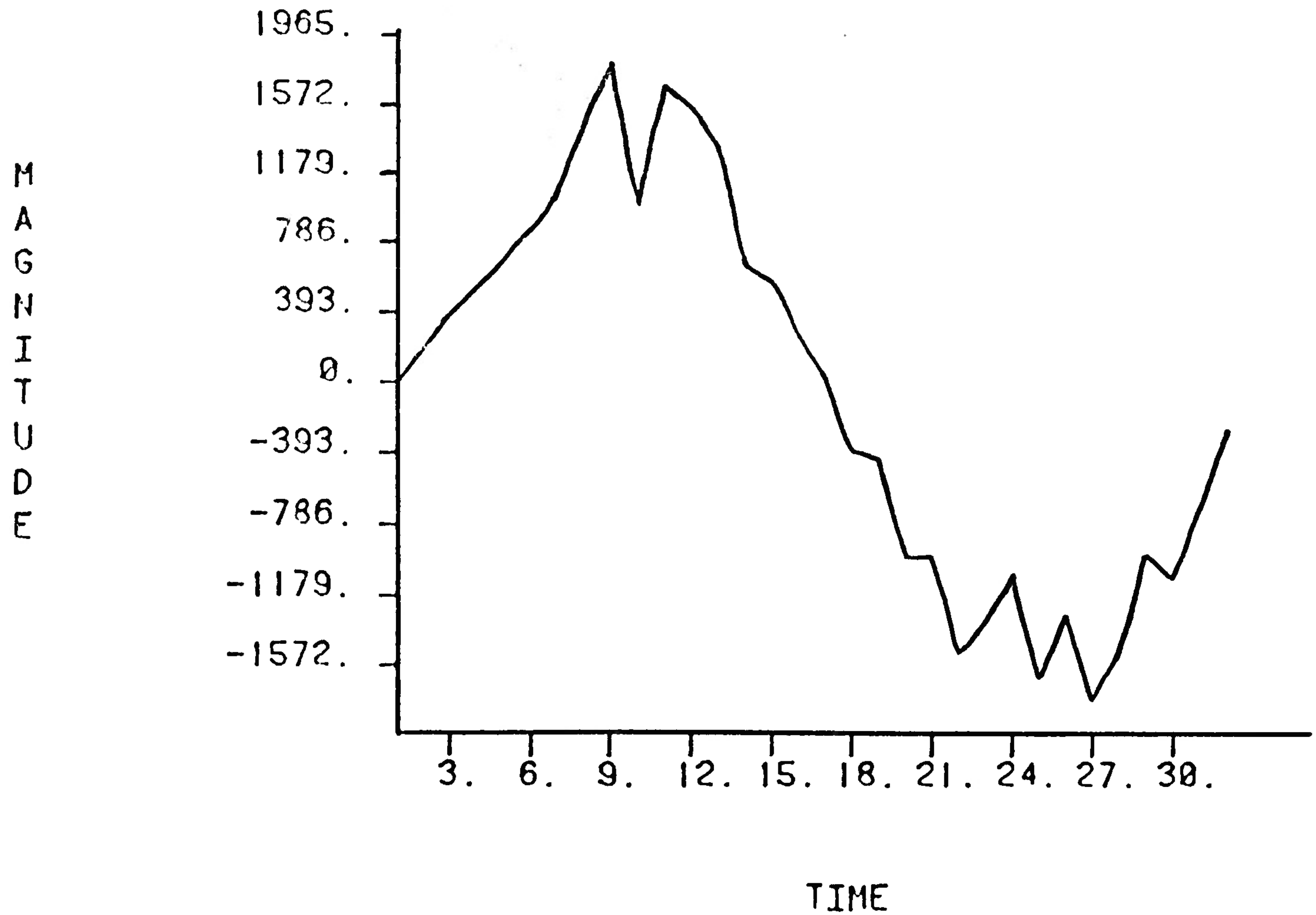


Figure 21. The sine wave with noise added to it.

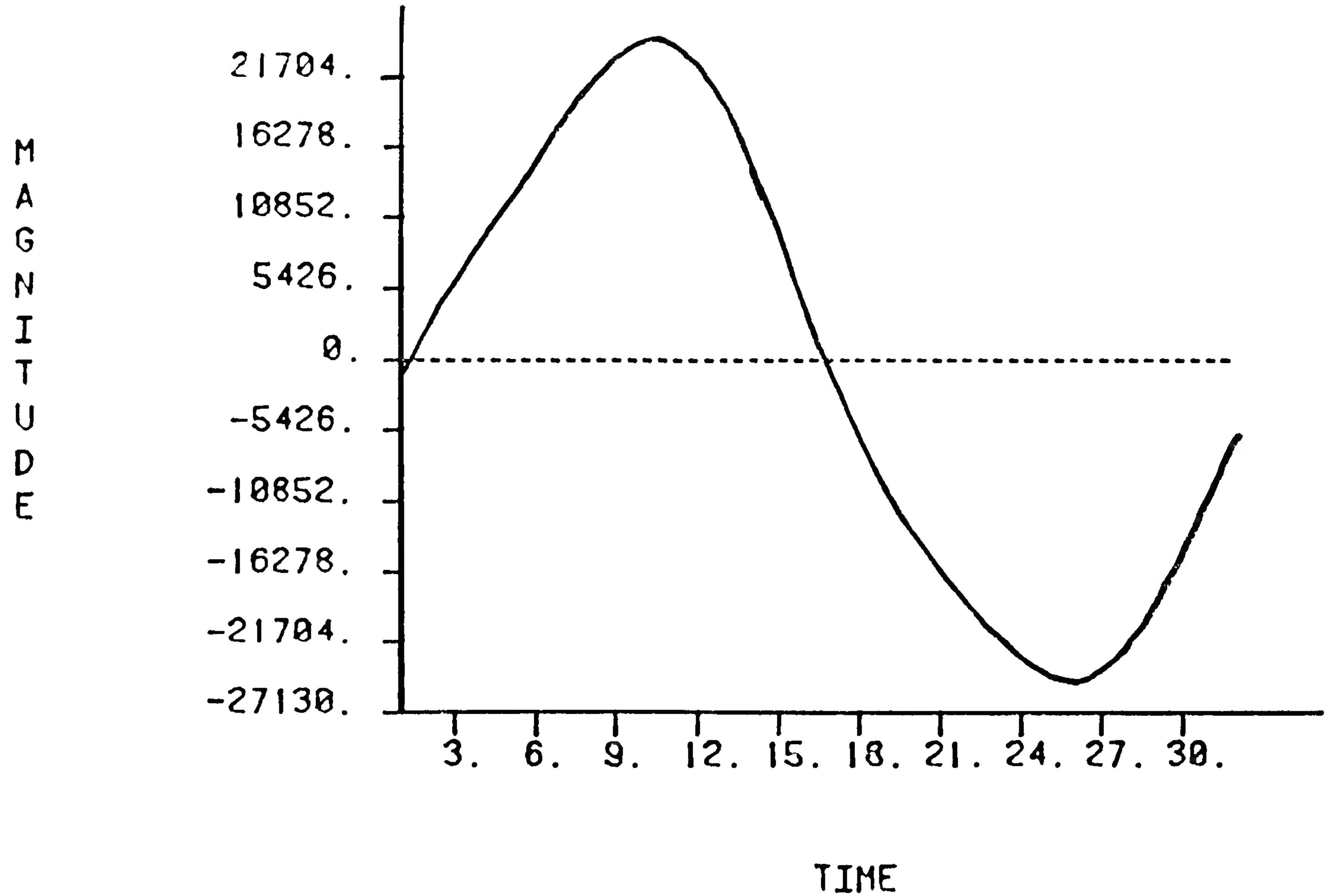


Figure 22. The results of the inverse transformation of the noisy spectrum after the optimal zero filling range.

the noisy spectrum after the optimal zero filling range.

Graphics. Graphics display would allow on-line monitoring of data acquisition plus it would greatly facilitate data interpretation. Each computer system in the LDS has a graphics display device. The resolution on the APPLE II system device is 192 by 280 pixels, for the IBM PC system it is 640 by 200 pixels (or 320 by 200 for color), and the LSI-11 system has a resolution of 512 by 250 pixels for the ADM terminal and 640 by 480 for the Tektronix 4025 terminal. In addition, the APPLE II and IBM PC systems have color graphics capability. Graphics hardcopy devices for the LDS includes an APPLE II silentype printer, Tektronix 4662 digital plotter, IDS 440 Paper Tiger printer, and a X-Y plotter.

The routine XYPLOT outputs the data to the X-Y plotter as illustrated in Figure 23. Figures 24, 25 and 26 illustrate output on the 4662 digital plotter. Data were outputted to the digital plotter by the program AGRAPH (see APPENDIX).

Communication. Each computer in the LDS network has the capability of communicating with any other computer and its peripherals. This allows the computers to share the larger, more expensive and reliable peripherals, thus eliminating the duplication of such peripherals.

Pipes are one method that enables different computers

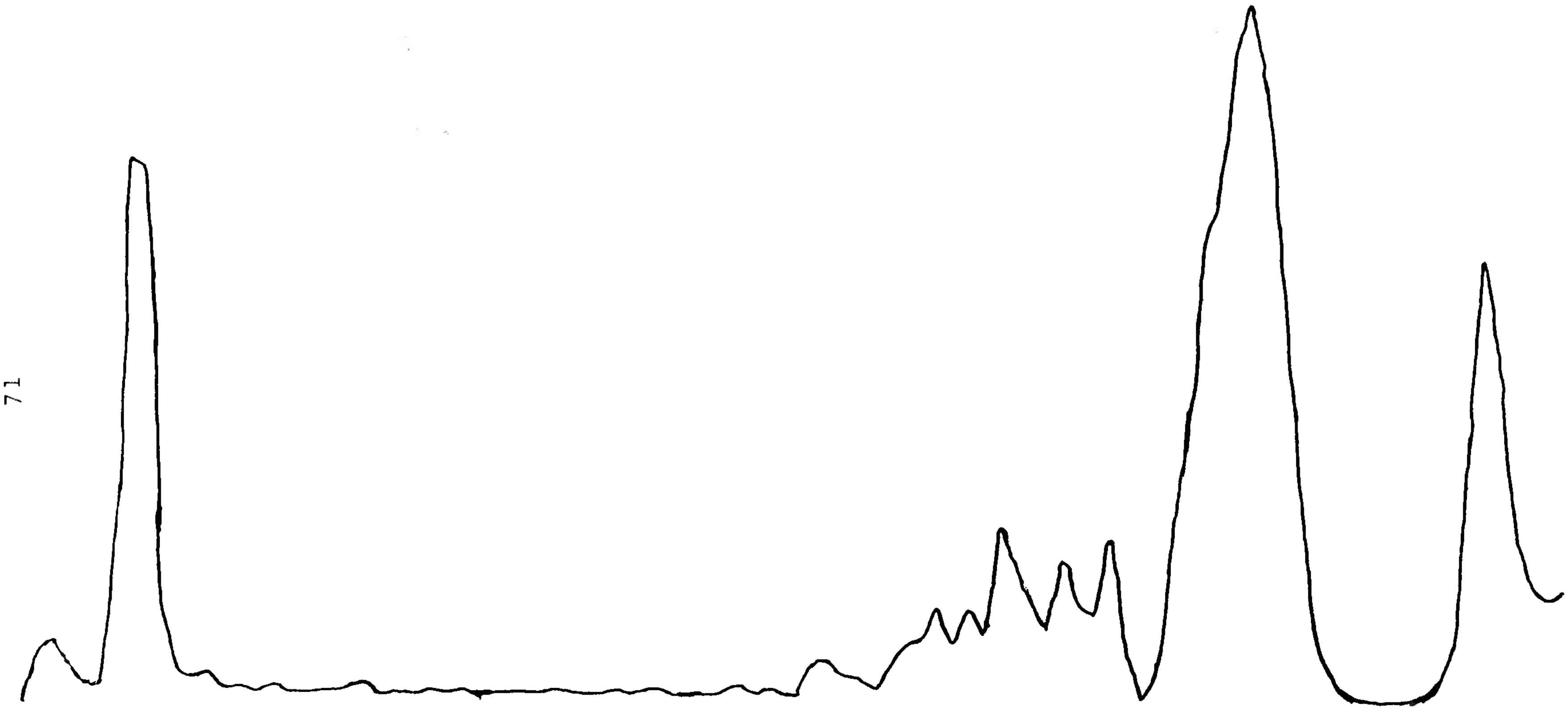


Figure 23. Computer simulated spectrum output to the X-Y plotter by the routine XYPLOT.

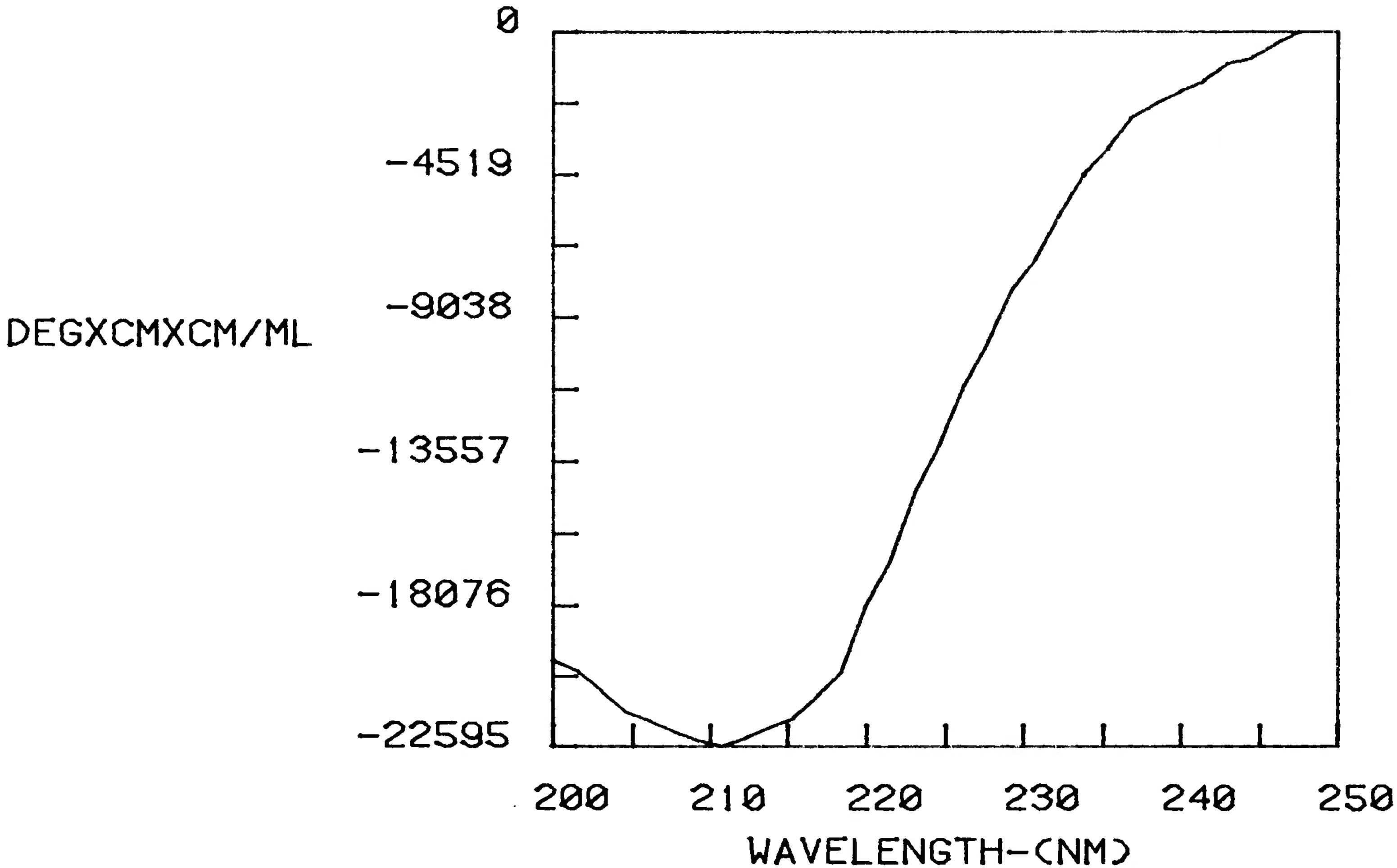


Figure 24. A circular dichroism spectrum output on a 4662 digital plotter by the program AGRAPH.

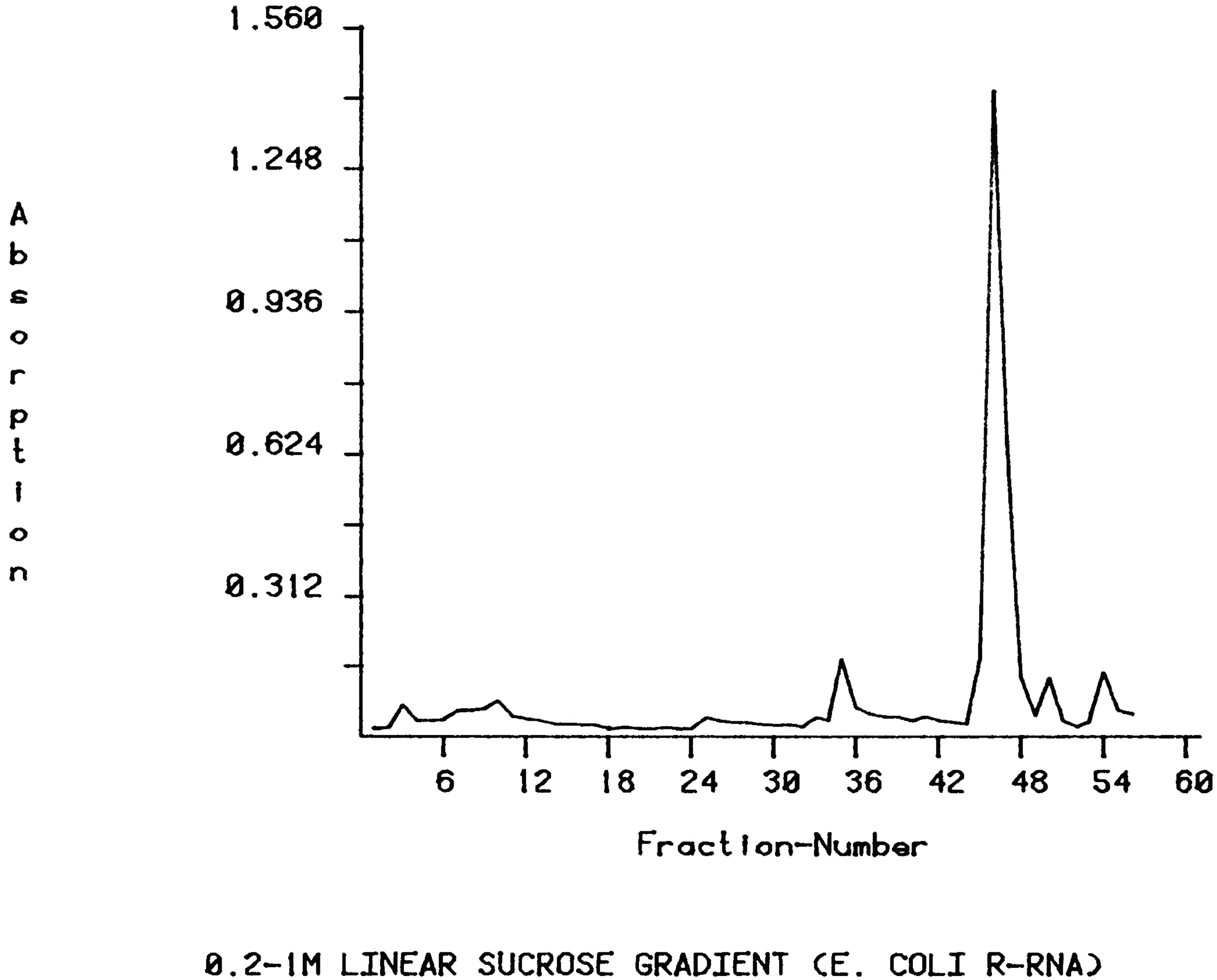


Figure 25. A plot of absorption versus fraction-number using the program AGRAPH.

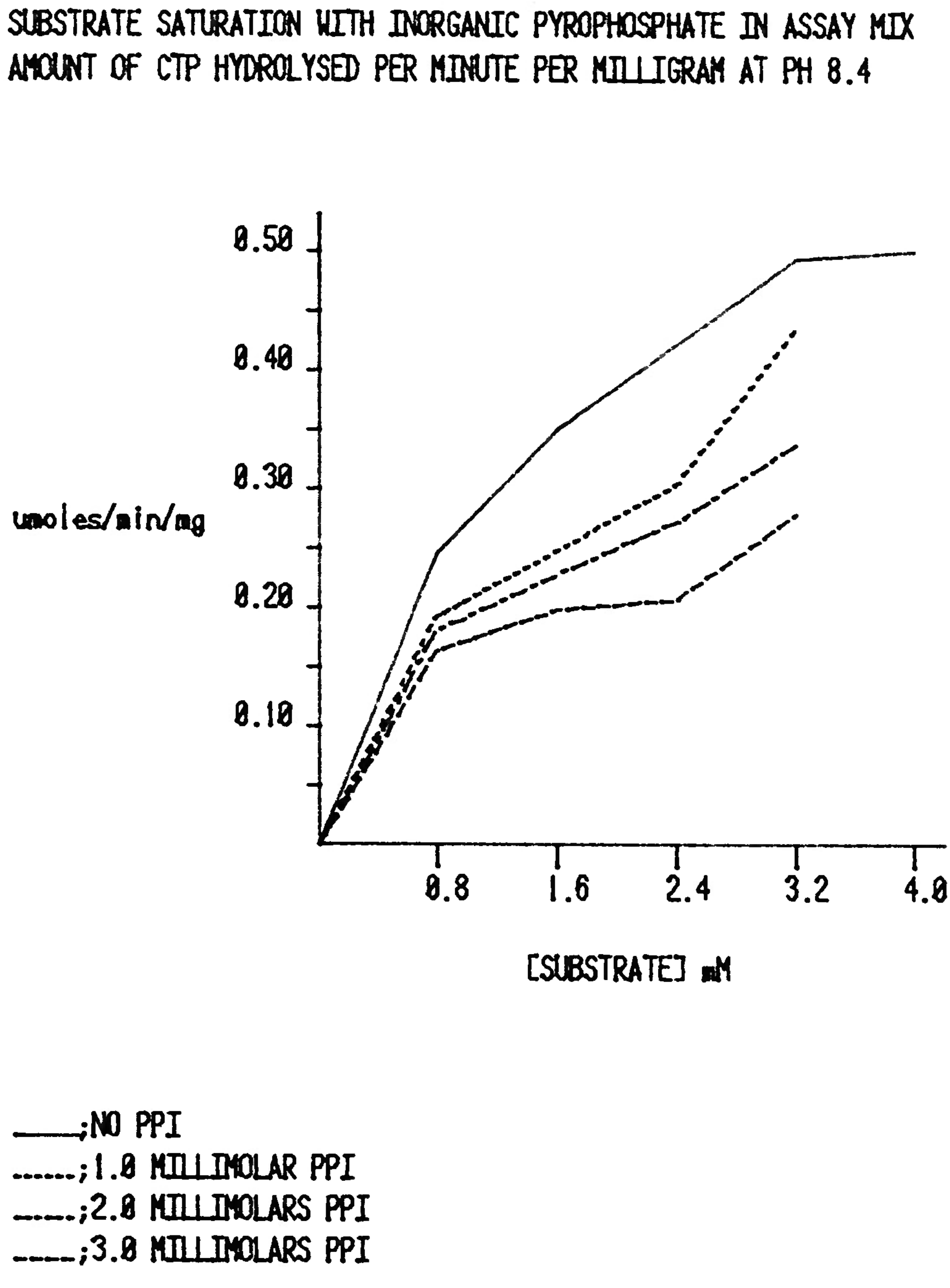


Figure 26. A plot of several curves using the program AGRAPH.

to communicate with each other or share common peripherals. The program SPOOL from Corvus Inc. uses Pipes. This program sends a file with a designated Pipe name to the Pipe area of the disk. Once in the Pipe area, any computer system in the LDS network can retrieve that Pipe by using the "D(espooler)" option in the SPOOL program. In addition to the Pipes, computers connected to the LDS via the Omninet transporter can communicate with each other using the transporter commands. The following routines were written to utilize the transporter commands:

SENDM ( )-Send message command directs the transporter to send a message to the indicated destination host and destination socket.

SETUP ( )-Setup receive command prepares a socket for the receiving of a single message.

ENDREC ( )-End receive command tells the transporter to release the specified socket.

INTRAN ( )-Initialize transporter command initializes the transporter as in a hardware reset or a power-up.

WHOAMI ( )- Who am I command returns the transporter's device address.

ECHO ( )-Echo command is used to verify the presence of another network device.

Each of these routines was written in the respective computer systems assembly language. Listings of these routines for the 6502 (APPLE II) assembly language and MACRO-11 are given in the APPENDIX.

VI. CONCULSION

The LDS design of a distributed local area network of microcomputers with the chemical research instruments interfaced to the microcomputers achieved the purpose of this research project of implementing a LDS for data acquisition and analysis. The Corvus Constellation and OMNINET schemes fit the requirements for an inexpensive, relatively high performance local area computer network. This LAN is quite flexible and new systems can be easily added as they are developed. With a powerful intercomputer communication protocol, the system allows information to be stored as a part of a created database. The flow of information to and from the database is controlled by the LDMS. Graphics and display programs help facilitate the interpretation of the data. Digital filtering programs enhanced the S/N ratio of noisy data. Furthermore, extensive software development can be done on the system without adversely affecting other functions. Thus, this relatively inexpensive system permits efficient resource and data sharing that is needed in the research laboratory.

REFERENCES

1. Frazer, J. W.; Rigdon, L. P.; Brand, H. R.; Pomernacki, C. L. Anal. Chem., 1979, 51, 1739.

2. Kryskow, J. M.; Miller, C. K. Computer Design, 1981, 20, 22.

3. Dessy, R. E. Anal. Chem., 1982, 54, 1167A.

4. Freeman, H. A. Computer, 1980, 13, 7.

5. Thornton, J. E. Computer, 1980, 13, 10.

6. Binkley, D. P.; Dessy, R. E. J. Chem. Ed., 1979, 56, 148.

7. Savitzky, A.; Golay, M. J. E. Anal. Chem., 1964, 36, 1627.

8. Phillips, J. B. Anal. Chem., 1980, 52, 468.

9. Griffiths, P. R. Appl. Spectrosc., 1975, 29, 11.

10. Binkley, D. P.; Dessy, R. E. Anal. Chem., 1980, 52, 1335.

11. "LSI-11 Plug-Compatible Disk Systems Reference Manual," Corvus Systems, Inc., San Jose, Cal., 1981.

12. "OMNINET Programmer's Guide," Corvus Systems, Inc., San Jose, Cal., 1982.

13. Jaeger, R. C. IEEE Mirco, 1982, 2, 20.

14. Strickland, C. M.S. Thesis, Atlanta University, Atlanta, Ga., 1981.

APPENDICES

```
C                        TERRY J. GREEN
C                        CHEMOMETRICS LABORATORY
C                        ATLANTA UNIVERSITY
C                        ATLANTA, GEORGIA
C
C                        AMLAB
C
      DIMENSION IARRAY(3005)
      IASIZE = 3000
5     CONTINUE
      TYPE 10
      ACCEPT 20, IANS
      IF (IANS .EQ. 'Q') GOTO 30
      IF (IANS .EQ. 'C') CALL CDACQU(IARRAY,IASIZE)
      IF (IANS .EQ. 'U') CALL UVACQU(IARRAY,IASIZE)
      IF (IANS .EQ. 'G') CALL GCACQU(IARRAY,IASIZE)
      IF (IANS .EQ. 'I') CALL IRACQU(IARRAY,IASIZE)
      GOTO 5
30    CONTINUE
C
C*************************** FORMAT STATMENTS ***************************
10    FORMAT (///1X,'DATA ACQUISITION FOR---',/1X,' C)IRCULAR DICHROISM',
     #  ' U)V-VISIBLE, G)AS CHROMATOGRAPH, I)NFRA RED, OR Q)UIT',2X,$)
20    FORMAT (A2)
C***********************************************************************
C
      END
```

```
C                        SUBROUTINE OUTDAT.FOR
C
C
C THIS SUBROUTINE OUTPUT AN ARRAY OF DATA TO A TERMINAL, LINE PRINTER, OR
C TO A DATA FILE.
C                                             ----------WRITTEN BY T. J. GREEN
C
C
        SUBROUTINE  OUTDAT (IARRAY, NPOINT, XSTART, XINCR)
        DIMENSION   IARRAY(NPOINT)
5       CONTINUE
        XVALUE=XSTART
        TYPE 10
        ACCEPT 20, IANS
        IF (IANS .EQ. 'T' .OR. IANS .EQ. 'P') GOTO 40
        IF (IANS .EQ. 'F' .OR. IANS .EQ. 'Q') GOTO 40
        TYPE 30
        GOTO 5
40      CONTINUE
        IF (IANS .EQ. 'Q') GOTO 70
        IF (IANS .EQ. 'F') GOTO 60
        IF (IANS .EQ. 'P') GOTO 50
        J = 0
        DO 45 I=1, NPOINT/10
        TYPE 68, XVALUE, (IARRAY(J),J=J+1,I*10)
        XVALUE=XVALUE-(XINCR*10)
        J = I * 10
45      CONTINUE
        GOTO 5
50      CONTINUE
        J = 0
        DO 55 I=1, NPOINT/10
        PRINT 68, XVALUE, (IARRAY(J),J=J+1,I*10)
        XVALUE=XVALUE-(XINCR*10)
        J = I * 10
55      CONTINUE
        GOTO 5
60      CONTINUE
        TYPE 65
        CALL ASSIGN (1,'XXXXXX.DAT',-1,'NEW',NC,1)
        TYPE 80
        WRITE (1,*) XVALUE, XINCR, (IARRAY(I),I=1,NPOINT)
        TYPE 90
        CALL CLOSE (1)
        GOTO 5
70      CONTINUE
C
C***************************** FORMAT STATEMENTS ************************
10      FORMAT (///1X,'OUTPUT DATA TO--- T)ERMINAL, P)RINTER, F)ILE, OR
     * Q)UIT',2X,$)
```

```
20      FORMAT (A2)
30      FORMAT (//////1X,'**********ILLEGAL OPTION**********',/////)
65      FORMAT (///1X,'INPUT THE FILE NAME',/)
68      FORMAT (F10.2,10(2X,I5))
80      FORMAT (//1X,'WRITING DATA TO FILE........................')
90      FORMAT (/1X,'WRITING COMPLETE')
C**********************************************************************
C
        RETURN
        END
```

```
C                            SUBROUTINE XYPLOT.FOR
C
C
C THIS SUBROUTINE OUTPUTS AN ARRAY OF POINTS TO THE OMNIGRAPHIC X-Y
C RECORDER
C                                          ----------WRITTEN BY T. J. GREEN
C
C
        SUBROUTINE XYPLOT(IARRAY,NPOINT,MAXNUM)
        DIMENSION IARRAY(NPOINT)
        IRANGE = NPOINT
        ISTART = 1
        INC = 1
        TYPE 10
        PAUSE 'HIT THE RETURN KEY TO CONTINUE'
15      CONTINUE
        TYPE 20
        ACCEPT 22, IANS1
        IF (IANS1 .EQ. 'E' .OR. IANS1 .EQ. 'W') GOTO 30
        IF (IANS1 .EQ. 'S' .OR. IANS1 .EQ. 'Q') GOTO 30
        TYPE 25
        GOTO 15
30      CONTINUE
        TYPE *
        TYPE *
        IF (IANS1 .EQ. 'Q') GOTO 70
        IF (IANS1 .EQ. 'S') GOTO 50
        IF (IANS1 .EQ. 'W') GOTO 50
        IRANGE = NPOINT
        ISTART = 1
        INC = 1
        GOTO 50
40      CONTINUE
        TYPE *, 'ENTER THE STARTING POINT NUMBER'
        ACCEPT *, ISTART
        TYPE *, 'ENTER THE NUMBER OF POINT TO BE PLOTTED'
        ACCEPT *, IRANGE
        TYPE *, 'ENTER THE INCREMENT'
        ACCEPT *, INC
50      CONTINUE
        INCR = INC*(MAXNUM/IRANGE)
        ISTART = ISTART-1
        CALL IPLOT(IARRAY,INCR,1)
        GOTO 15
70      CONTINUE
C
C*************************** FORMAT STATEMENTS *********************
10      FORMAT (///1X,'TURN ON THE OMNIGRAPHIC RECORDER AND SET THE',
     +   ' CONTROL',////)
20      FORMAT (//1X,'PLOT E)NTIRE SPECTRUM, W)INDOW, S)AME WINDOW,',
```

```
     +    ' OR Q)UIT',2X,$)
22      FORMAT (A2)
25      FORMAT (/////1X,'********** ILLEGAL OPTION **********',/////)
C*****************************************************************************
C
        RETURN
        END
```

```
        .TITLE  IPLOT.MAC
;
; THIS SUBROUTINE OUTPUT AN ARRAY OF NUMBER TO THE D/A BUFFER (CHANNEL A)
; ON THE DT2781 A/D BOARD (CHANNEL A=170404, B=170406).
;                 -----------WRITTEN BY T. J. GREEN
;
        .GLOBL  IPLOT
;
        .MCALL  .PRINT,.TTYIN
;
IPLOT:  TST     (R5)+
        MOV     (R5)+,R4
        MOV     @(R5)+,IFAC
        MOV     @(R5)+,IBUF
        BLE     RETURN
        CLR     R3
        CLR     R2
        CLR     @#170404
        CLR     @#170406
        .PRINT  #MSG1
        .PRINT  #MSG2
INLP:   .TTYIN
        CMPB    R0,#15
        BEQ     OUT
        BR      INLP
OUT:    .PRINT  #MSG3
        MOV     #1800.,AFAC
        SUB     NUM,AFAC
        MOV     #0.,R3
NEXT:   MOV     R3,@#170406
        MOV     (R4)+,@#170404
WAIT1:  MOV     #350.,COUNT
WAIT:   NOP
        DEC     COUNT
        BNE     WAIT
        INC     R2
        CMP     R2,IFAC
        BLT     WAIT1
        CLR     R2
        ADD     IFAC,R3
        CMP     R3,#1800.
        BLT     NEXT
        CLR     R0
        .PRINT  #MSG6
        .PRINT  #MSG6
        .PRINT  #MSG4
        .PRINT  #MSG6
        .PRINT  #MSG5
        .PRINT  #MSG2
INLP1:  .TTYIN
```

```
        CMPB    R0,#15
        BEQ     OUT1
        BR      INLP1
OUT1:   CLR     @#170404
        CLR     @#170406
;
RETURN: RTS     PC
AFAC:   .BLKW   1
IFAC:   .BLKW   1
IBUF:   .BLKW   1
COUNT:  .BLKW   1
NUM:    .BLKW   1
MSG1:   .ASCIZ  *FLIP THE RESET/SWEEP SWITCH ON THE PLOTTER TO SWEEP*
MSG2:   .ASCIZ  /HIT THE RETURN KEY TO CONTINUE/
MSG3:   .ASCIZ  /PLOTTING IN PROGRESS...................../
MSG4:   .ASCIZ  /PLOTTING DONE/
MSG5:   .ASCIZ  *FLIP THE RESET/SWEEP SWITCH TO RESET*
MSG6:   .ASCIZ  /   /
        .END
```

```
        .TITLE  DISP
;
; THIS ROUTINE DISPLAYS THE DATA ON AN OSCILLOSCOPE.
;
        .GLOBL  IDISP
        .MCALL  .PRINT .TTINR
;
JSW=44
;
IDISP:  TST     (R5)+
        MOV     (R5)+,IBUF
        MOV     @(R5)+,IFAC
        MOV     @(R5)+,NUM
        MOV     (R5)+,R0
        MOV     @(R5)+,DELAY
        BLE     RETURN
        MOV     R0,R5
REPEAT: JSR     PC,SCOPE
        CMP     (R5),#1
        BNE     REPEAT
        .PRINT  #MESG1
        JSR     PC,QUIT
RETURN: RTS     PC
;
;  SUBROUTINE SCOPE
SCOPE:  MOV     #-2048.,R3
        MOV     NUM,COUNT
        MOV     IBUF,R4
NEXT:   MOV     (R4)+,YVAL
        ADD     #-2048.,YVAL
        MOV     YVAL,@#170404
        MOV     R3,@#170406
        ADD     IFAC,R3
        MOV     DELAY,R2
STIME:  SOB     R2,STIME
        DEC     COUNT
        BNE     NEXT
        RTS     PC
;
;  SUBROUTNIE QUIT
QUIT:   MOV     (SP)+,R1
        BIS     #100,@#JSW
NOCHAR: .TTINR
        BCC     OUT
        JSR     PC,SCOPE
        BR      NOCHAR
OUT:    BIC     #100,JSW
        MOV     R1,-(SP)
        RTS     PC
;
IBUF:   .BLKW   1
```

```
IF4C:   .BLKW   1
CMF:    .BLKW   1
COUNT:  .BLKW   1
DELAY:  .BLKW   1
NUM:    .BLKW   1
YVAL:   .BLKW   1
MESG1:  .ASCII  /DATA ACQUISITION COMPLETE-HIT RETURN TO CONTINUE /
        .END
```

```
        .TITLE  UVCON
;
; THIS ROUTINE ACQUIRES THE DATA POINTS FROM THE CARY 17.
;
        .GLOBL  IUVCON
        .MCALL  .PRINT,.TTINR,.TTYIN,.MTPS
;
JSW=44
ADCSR=170400
ADDBR=170402
DABUFA=170404
DABUFB=170406
RTCCSR=170420
RTCBPR=170422
ADVEC=400
ERVEC=404
CLKVEC=440
;
IUVCON: TST     (R5)+
        MOV     (R5)+,IBUF
        MOV     @(R5)+,IFAC
        MOV     @(R5)+,NUM
        MOV     @(R5),DELAY
        BLE     RETURN
        MOV     #400,R1
        MOV     IBUF,R3
        MOV     NUM,DCOUNT
        MOV     #-16.,RTCBPR
        MOV     #140,RTCCSR
        MOV     #OVFINT,CLKVEC
        MOV     #ADINT,ADVEC
        MOV     #6,STCH
        SWAB    STCH
        BIC     #377,STCH
        ADD     #120,STCH
        .MTPS   #0
        TST     ADDBR
        .PRINT  #MESG0
        BIS     #10100,@#JSW
 INLOOP: .TTYIN
        CMPB    R0,#12
        BNE     INLOOP
        MOV     STCH,ADCSR
        .PRINT  #MESG2
 REPEAT: CALL   DISP
        CMP     DCOUNT,#0
        BGT     REPEAT
        CLR     ADCSR
        .MTPS   #340
        .PRINT  #MESG1
 NOCHAR: .TTINR
```

```
        BCC     OUT
        CALL    DISP
        BR      NOCHAR
OUT:    BIC     #10100,@#JSW
        MOV     #-2048.,DABUFA
        MOV     #-2048.,DABUFB
RETURN: RTS     PC
;
;    A/D DONE INTERRUPT HANDLER
ADINT:  MOV     ADDBR,(R3)+
        INC     RTCCSR
        XOR     R1,#ADCSR
        DEC     DCOUNT
        RTI
;
;    RTC OVERFLOW INTERRUPT HANDLER
OVFINT: INCB    ADCSR
        RTI
;
;  SUBROUTINE DISP
DISP:   MOV     #-2048.,XVAL
        MOV     NUM,COUNT
        MOV     IBUF,R4
NEXT:   MOV     (R4)+,YVAL
        ADD     #-2048.,YVAL
        MOV     YVAL,DABUFA
        MOV     XVAL,DABUFB
        ADD     IFAC,XVAL
        MOV     DELAY,R2
STIME:  SOB     R2,STIME
        DEC     COUNT
        BNE     NEXT
        RETURN
;
;
IBUF:   .BLKW   1
IFAC:   .BLKW   1
CLKC:   .BLKW   1
COUNT:  .BLKW   1
DCOUNT: .BLKW   1
DELAY:  .BLKW   1
NUM:    .BLKW   1
STCH:   .BLKW   1
XVAL:   .BLKW   1
YVAL:   .BLKW   1
MESG0:  .ASCIZ  /HIT THE RETURN KEY TO START ------------/
MESG2:  .ASCIZ  /DATA ACQUISITION IN PROGRESS ................/
MESG1:  .ASCII  /DATA ACQUISITION COMPLETE --- HIT RETURN TO CONTINUE /
        .END
```

```
C                         TERRY J. GREEN
C                         CHEMOMETRICS LABORATORY
C                         ATLANTA UNIVERSITY
C                         ATLANTA, GEORGIA
C
      DIMENSION PNTS(1500), NUM(10), XST(10), XINC(10), YSCAL(10)
      LOGICAL*1 FNAME(12)
      ISZ = 1500
      MAXBUF=ISZ
      MAXFIL=10
      NFILE=0
      NPTS=1
5     IF (ITTOUR(26) .NE. 0) GOTO 5
      TYPE 10
      TYPE 11
30    CONTINUE
      NFILE=NFILE+1
      CALL LINEF
      TYPE 12
      CALL IPOKE('44,'10000 .OR. IPEEK('44))
32    ICH = ITTINR()
      IF (ICH .EQ. 70) GOTO 45
      IF (ICH .EQ. 84) GOTO 35
      GOTO 32
35    CONTINUE
      CALL IPOKE ('44,'10000 .XOR. IPEEK('44))
      CALL KEYIN(PNTS,ISZ,MAXBUF,NPTS,NUM(NFILE),XST(NFILE),XINC(NFILE))
      GOTO 55
45    CONTINUE
      CALL IPOKE ('44,'10000 .XOR. IPEEK('44))
47    IF (ITTOUR(26) .NE. 0) GOTO 47
      TYPE *,'ENTER THE DATA FILE NAME  '
      ACCEPT 200,FNAME
      CALL ASSIGN (1,FNAME,0,'RDO',NC,1)
      TYPE 48,FNAME
      READ (1,*) NUM(NFILE)
      CALL CLOSE (1)
      NUM1=NUM(NFILE)
      IF (NUM1 .GT. MAXBUF) GOTO 60
      ITOF=NPTS+NUM1-1
      CALL ASSIGN (1,FNAME,0,'RDO',NC,1)
      READ (1,*) IDUMMY,XST(NFILE),XINC(NFILE),YSCAL(NFILE),
     1  (PNTS(I),I=NPTS,ITOF)
      CALL CLOSE (1)
      DO 50 I=NPTS,ITOF
      PNTS(I)=PNTS(I)*YSCAL(NFILE)
50    CONTINUE
      NPTS=NPTS+NUM1
55    CONTINUE
      MAXBUF=MAXBUF-(NPTS-1)
      IF (MAXBUF .LT. 2) GOTO 65
      IF (NFILE .GE. MAXFIL) GOTO 65
```

```
        CALL LINEF
        TYPE 57
        CALL IPOKE('44,'10000 .OR. IPEEK('44))
58      IANS = ITTINR()
        IF (IANS .EQ. 78) GOTO 65
        IF (IANS .EQ. 89) GOTO 30
        GOTO 58
60      CONTINUE
        TYPE *, 'SORRY---THERE IS NOT ENOUGH BUFFER SPACE'
        IF (NFILE .EQ. 1) STOP
63      CONTINUE
        TYPE *, 'DO YOU WANT TO CONTINUE (Y/N)'
        CALL IPOKE ('44,'10000 .OR. IPEEK('44))
64      IAN = ITTINR()
        IF (IAN .EQ. 78) STOP
        IF (IAN .EQ. 89) GOTO 65
        GOTO 64
65      CONTINUE
        XMAX1=((NUM(1)*XINC(1))-XINC(1))+XST(1)
        IF (NFILE .GT. 1) GOTO 70
        XMIN1=XST(1)
        GOTO 80
70      CONTINUE
        CALL MINMAX(XST,NFILE,XMIN1,TMAX1)
        DO 75 I=2,NFILE
        DMAX=((NUM(NFILE)*XINC(NFILE))-XINC(NFILE))+XST(NFIILE)
        IF (DMAX .GT. XMAX1) XMAX1=DMAX
75      CONTINUE
80      CONTINUE
        NPTS=NPTS-1
        CALL CURVE(PNTS,NPTS,NUM,NFILE,XST,XINC,XMIN1,XMAX1)
C
C************************ FORMAT STATEMENTS ************************
10      FORMAT (20X,'*** WELCOME TO AGRAPH ***'/40X,
     1                     '----------WRITTEN BY T. J. GREEN',//,
     2   ' THIS PROGRAM PLOTS SETS OF X-Y DATA FROM A DATA FILE OR',
     3   ' THE DATA CAN BE',/,' ENTERED VIA THE TERMINAL KEYBOARD.',
     4   '    DATA IN A FILE MUST BE IN THE FOL-',/' LOWING FORMAT:')
11      FORMAT(/,'            1) NUMBER OF Y-(DATA) POINTS',/,
     1   '           2) INITIAL X-VALUE',/,
     2   '           3) X-INCREMENT',/,
     3   '           4) Y-SCALE FACTOR',/
     4   '           5) Y-(DATA) POINTS')
12      FORMAT (' DATA FROM A F)ILE OR T)ERMINAL',$,1X)
57      FORMAT (1X,'DO YOU WANT TO ENTER MORE DATA (Y/N)?',$,1X)
49      FORMAT(' READING DATA FROM FILE.....',12A1,$,)
200     FORMAT(12A1)
C*****************************************************************
C
1000    IF (ITTOUR(25) .NE. 0) GOTO 1000
1005    IF (ITTOUR(24) .NE. 0) GOTO 1005
        STOP
```

```
      SUBROUTINE KEYIN(YPTS,ISIZE,MAXNUM,IST,NUM,XST,XINC)
      DIMENSION YPTS(ISIZE)
      NTL = 10
      IJ=1
5     IF (ITTOUR(26) .NE. 0) GOTO 5
      TYPE *,'DATA FROM THE KEYBOARD ........................'
      TYPE 10
      ACCEPT  *, XST
      TYPE 15
      ACCEPT  *, XINC
17    TYPE 20
      ACCEPT *, NUM
      IF (NUM .GT. MAXNUM) GOTO 1000
18    IF (ITTOUR(26) .NE. 0) GOTO 18
      TYPE 25, NUM
      ACCEPT *, (YPTS(I),I=IST,NUM+IST-1)
      TYPE 28
      CALL IPOKE('44,'10000 .OR. IPEEK('44))
40    ICHAR=ITTINR ()
      IF (ICHAR .EQ. 89) GOTO 45
      IF (ICHAR .EQ. 78) GOTO 300
      GOTO 40
45    CONTINUE
47    IF (ITTOUR(26) .NE. 0) GOTO 47
      TYPE 46
9000  CALL IPOKE('44,'10000 .OR. IPEEK('44))
      ICHAR = -1
      TYPE 30
48    ICHAR = ITTINR()
      IF (ICHAR .EQ. 76) GOTO 49
      IF (ICHAR .EQ. 67) GOTO 65
      IF (ICHAR .EQ. 65) GOTO 80
      IF (ICHAR .EQ. 73) GOTO 90
      IF (ICHAR .EQ. 82) GOTO 120
      IF (ICHAR .EQ. 81) GOTO 300
      GOTO 48
49    CONTINUE
55    IF (ITTOUR(26) .NE. 0) GOTO 55
      TYPE 46
      TYPE 32
      IF (NUM .LT. NTL) NLINE = NUM
      TYPE *,'POINT NUMBER        Y-VALUES'
      DO 60 I=1,NLINE
      IF (IJ .GT. NUM .AND. I .NE. 1) TYPE 200
      IF (IJ .GT. NUM) IJ = 1
      TYPE *, IJ,'           ', YPTS(IJ+IST-1)
      IJ = IJ +1
60    CONTINUE
      TYPE *
      GOTO 48
65    CONTINUE
      CALL IPOKE('44,'10000 .XOR. IPEEK('44))
```

```
63      TYPE 68
        ACCEPT *, IC
        IF (IC .LT. 1 .OR. IC .GT. NUM) GOTO 70
        TYPE 72
        ACCEPT *, YPTS(IC+IST-1)
        IJ = IC
        GOTO 9000
70      CONTINUE
        TYPE 74, NUM
        GOTO 63
80      CONTINUE
        CALL IPOKE('44,'10000 .XOR. IPEEK('44))
        IF (NUM .GE. MAXNUM) GOTO 2000
        IAD = MAXNUM - NUM
82      TYPE 84
        ACCEPT *, IADD
        IF ((IADD+NUM) .GT. MAXNUM) GOTO 3000
        IJ = NUM+1
        IOF=NUM+IST
        NUM=NUM+IADD
        TYPE 86, IADD
        ACCEPT *, (YPTS(I),I=IOF,NUM+IST-1)
        GOTO 9000
90      CONTINUE
        CALL IPOKE('44,'10000 .XOR. IPEEK('44))
92      TYPE 94
        ACCEPT *, INS
        IOFI = (IST+INS)-1
        IF (INS .LT. 1) GOTO 4000
        IF (INS .GT. NUM) GOTO 5000
        TYPE 96
        ACCEPT *, TEMP
        IT = 1
        NUM = NUM + 1
        ITOP = NUM+IST-1
        DO 105 I=IOFI,ITOP
        YPTS(ITOP-IT+1) = YPTS(ITOP-IT)
        IT=IT+1
105     CONTINUE
        YPTS(IOFI)=TEMP
        IJ = INS
        GOTO 9000
120     CONTINUE
        CALL IPOKE('44,'10000 .XOR. IPEEK('44))
122     TYPE 125
        ACCEPT *,IRM
        IOFR = (IST+IRM)-1
        IF (IRM .LT. 1) GOTO 4000
        IF (IRM .GT. NUM) GOTO 5000
        NUM = NUM - 1
        ITOP = NUM+IST-1
        DO 130 I=IOFR,ITOP
```

```
      YPTS(I) = YPTS(I+1)
130   CONTINUE
      IJ = IRM
      GOTO 9000
1000  CONTINUE
      TYPE 16, MAXNUM
      GOTO 17
2000  CONTINUE
      TYPE *,'NO MORE VALUES CAN BE ADDED'
      GOTO 9000
3000  CONTINUE
      TYPE 89, IAD
      GOTO 9000
4000  CONTINUE
      TYPE *,'POINT NUMBER TOO SMALL!!!!!!!!'
      GOTO 9000
5000  CONTINUE
      TYPE *,'POINT NUMBER TOO LARGE!!!!!!!!'
      GOTO 9000
C                       FORMAT STATEMENTS
10    FORMAT (//1X,'ENTER THE X-INITIAL VALUE')
15    FORMAT (1X,'ENTER THE X-INCREMENT')
16    FORMAT (' THE MAXIMUM NUMBER TO ENTER IS ',I5)
20    FORMAT (1X,'ENTER THE NUMBER OF Y-POINTS TO BE INPUTTED')
25    FORMAT (1X,'PLEASE ENTER ',I6,'   Y-VALUES')
28    FORMAT (//,' DO YOU WANT TO EDIT THE DATA (Y/N)',$,1X)
30    FORMAT (' L)IST C)HANGE A)DD I)NSERT R)EMOVE OR Q)UIT',/)
32    FORMAT (' L)IST C)HANGE A)DD I)NSERT R)EMOVE OR Q)UIT')
46    FORMAT(' EDITING ..................................')
68    FORMAT(' ENTER THE POINT # TO BE CHANGED')
72    FORMAT(' ENTER THE NEW VALUE')
74    FORMAT(' THE POINT # MUST BE IN THE RANGE OF 1 AND ',I6)
84    FORMAT(' ENTER THE NUMBER OF POINT(S) TO BE ADDED')
86    FORMAT(' ENTER ',I6,' VALUES')
89    FORMAT(' ONLY ',I5,' VALUES CAN BE ADDED')
94    FORMAT(' ENTER THE POINT # FOR THE VALUE TO BE INSERTED')
96    FORMAT(' ENTER THE NEW Y-VALUE')
125   FORMAT(' ENTER THE POINT # FOR THE VALUE TO BE REMOVED')
200   FORMAT('*******************************************')
305   FORMAT(' DO YOU WANT TO SAVE THE DATA (Y/N)',$,1X)
C
300   CONTINUE
      CALL LINEF
      TYPE 305
310   IANS = ITTINR()
      IF (IANS .EQ. 78) GOTO 320
      IF (IANS .EQ. 89) GOTO 315
      GOTO 310
315   CONTINUE
      CALL IPOKE("44,"10000 .XOR. IPEEK("44))
      CALL SFILE(YPTS,NUM,XST,XINC)
      GOTO 325
```

```
320     CONTINUE
        CALL IPOKE('44,'10000 .XOR. IPEEK('44))
325     IST = IST+NUM
        RETURN
        END

        SUBROUTINE SFILE (YPTS,NUM,XST,XINC)
        DIMENSION YPTS(NUM)
        YSCALE=1.0
        CALL LINEF
        TYPE 10
10      FORMAT(' ENTER THE FILENAME (6 CHARACTERS MAX)',$,1X)
        CALL ASSIGN (1,'XXXXXX.DAT',-1,'NEW',NC,1)
        WRITE (1,*) NUM,XST,XINC,YSCALE,(YPTS(I),I=1,NUM)
        CALL CLOSE (1)
        RETURN
        END

        SUBROUTINE LINEF
10      IF (ITTOUR (13) .NE. 0) GOTO 10
20      IF (ITTOUR (10) .NE. 0) GOTO 20
        RETURN
        END
```

```
C                          SUBROUTINE ASCODE.FOR
C
C
C THIS SUBROUTINE RETURNS THE ASCII CODE OF A NUMBER
C                                   ----------WRITTEN BY T. J. GREEN
C
C
        SUBROUTINE ASCODE (YNUM,ITRANS,J)
        DOUBLE PRECISION YNUM1, TENTH, ONE, TEN, ZERO
        DOUBLE PRECISION ADD, TMULTY, TRANGE, BRANGE, XN, YN, ZN
        DIMENSION ICHAR(12), ITRANS(15)
        DATA ICHAR/49,50,51,52,53,54,55,56,57,48,46,45/
        ZERO=0.0
        TENTH=0.10000
        ONE=1.00000
        TEN=10.00000
        ADD=0.00090
C THE MAXIMUM NUMBER IS 999,999,999 AND THE MINIMUM NO. IS 0.001
        TMULTY=1.0D-10
        TRANGE=1.0D+10
        BRANGE=1.0D-04
        N1=0
        N2=10
        NDEC=2
C DETERMINE WHETHER THE NUMBER IS OUT OF RANGE
        IF (DABS(YNUM) .GE. TRANGE) GOTO 5
        IF (DABS(YNUM) .LE. BRANGE .AND. YNUM .NE. ZERO) GOTO 5
        GOTO 15
5       TYPE 10
        STOP
C DETERMINE WHETHER THE ABSOLUTE VALUE OF A NUMBER IS
C LESS THAN ONE
15      IF (DABS(YNUM) .LT. ONE .OR. YNUM .EQ. ZERO) GOTO 40
C DETERMINE THE ASCII CODE OF A NUMBER GREATER OR EQUAL TO ONE
        L=2
        J=14
20      YNUM1=(DABS(YNUM))*TMULTY
        XN=(DMOD(YNUM1,ONE))*TEN
        N=IDINT(XN)
        IF (N .EQ. N1) GOTO 25
        GOTO 30
25      TMULTY=TMULTY*TEN
        J=J-1
        GOTO 20
30      ITRANS(L)=ICHAR(N)
        L=L+1
        ICOUNT=J-1
        IJ=J-NDEC
        DO 35 I=1, ICOUNT
        IF (L .EQ. IJ) L=IJ+1
        YN=(DMOD(XN,ONE))*TEN
        YN=YN+ADD
```

```
      ZN=(DMOD(YN,ONE))*TEN
      N3=IDINT(ZN)
      XN=YN-ADD
      IF (N3 .EQ. N1) YN=YN+(YN*TENTH)
      N=IDINT(YN)
      IF (N .EQ. N1 .OR. N .GE. N2) N=N2
      ITRANS(L)=ICHAR(N)
      L=L+1
35    CONTINUE
      ITRANS(IJ)=ICHAR(11)
      GOTO 50
C DETERMINE THE ASCII CODE OF A NUMBER LESS THAN ONE
40    L=2
      J=5
      ITRANS(L)=ICHAR(10)
      L=L+1
      ITRANS(L)=ICHAR(11)
      L=L+1
      ICOUNT=J-2
      XN=YNUM
      DO 45 I=1, ICOUNT
      YN=(DMOD(XN,ONE))*TEN
      YN=YN+ADD
      ZN=(DMOD(XN,ONE))*TEN
      N3=IDINT(ZN)
      XN=YN-ADD
      IF (N3 .EQ. N1) YN=YN+(YN*TENTH)
      N=IDINT(YN)
      IF (N3 .EQ. N1 .OR. N .GE. N2) N=N2
      ITRANS(L)=ICHAR(N)
      L=L+1
45    CONTINUE
50    CONTINUE
      IF (YNUM .GE. ZERO) GOTO 55
      ITRANS(1)=ICHAR(12)
      J=J+1
      GOTO 65
55    CONTINUE
      DO 60 I=1,J
      ITRANS(I)=ITRANS(I+1)
60    CONTINUE
65    CONTINUE
C
C************************ FORMAT STATEMENT *************************
10    FORMAT (1X,'OUT OF RANGE----------OUT OF RANGE')
C******************************************************************
C
      RETURN
      END
```

```
C                         SUBROUNTINE HEADGR.FOR
C
C
C THIS SUBROUTINE DRAWS SIX DIFFERENT GRAPH AXIS
C                                       ----------WRITTEN BY T. J. GREEN
C
C
      SUBROUTINE HEADGR
      CALL MOVABS (25,600)
      CALL DRWABS (25,500)
      CALL DRWABS (125,500)
      CALL DRWABS (125,600)
      CALL DRWABS (25,600)
      CALL MOVABS (25,575)
      CALL DRWREL (10,0)
      CALL MOVABS (25,550)
      CALL DRWREL (10,0)
      CALL MOVABS (25,525)
      CALL DRWREL (10,0)
      CALL MOVABS (50,500)
      CALL DRWREL (0,10)
      CALL MOVABS (75,500)
      CALL DRWREL (0,10)
      CALL MOVABS (100,500)
      CALL DRWREL (0,10)
      CALL MOVABS (200,600)
      CALL DRWABS (200,500)
      CALL DRWABS (300,500)
      CALL DRWABS (300,600)
      CALL MOVABS (190,600)
      CALL DRWREL (10,0)
      CALL MOVABS (190,575)
      CALL DRWREL (10,0)
      CALL MOVABS (190,550)
      CALL DRWREL (10,0)
      CALL MOVABS (190,525)
      CALL DRWREL (10,0)
      CALL MOVABS (225,490)
      CALL DRWREL (0,10)
      CALL MOVABS (250,490)
      CALL DRWREL (0,10)
      CALL MOVABS (275,490)
      CALL DRWREL (0,10)
      CALL MOVABS (310,600)
      CALL DRWREL (-10,0)
      CALL MOVABS (310,575)
      CALL DRWREL (-10,0)
      CALL MOVABS (310,550)
      CALL DRWREL (-10,0)
      CALL MOVABS (310,525)
      CALL DRWREL (-10,0)
      CALL MOVABS (375,600)
```

```
CALL DRWABS (375,500)
CALL DRWABS (475,500)
CALL MOVABS (365,600)
CALL DRWREL (10,0)
CALL MOVABS (365,575)
CALL DRWREL (10,0)
CALL MOVABS (365,550)
CALL DRWREL (10,0)
CALL MOVABS (365,525)
CALL DRWREL (10,0)
CALL MOVABS (400,490)
CALL DRWREL (0,10)
CALL MOVABS (425,490)
CALL DRWREL (0,10)
CALL MOVABS (450,490)
CALL DRWREL (0,10)
CALL MOVABS (475,490)
CALL DRWREL (0,10)
CALL MOVABS (600,600)
CALL DRWABS (600,500)
CALL MOVABS (590,600)
CALL DRWREL (20,0)
CALL MOVABS (590,575)
CALL DRWREL (20,0)
CALL MOVABS (590,525)
CALL DRWREL (20,0)
CALL MOVABS (590,500)
CALL DRWREL (20,0)
CALL MOVABS (550,540)
CALL DRWREL (0,20)
CALL MOVABS (575,540)
CALL DRWREL (0,20)
CALL MOVABS (625,540)
CALL DRWREL (0,20)
CALL MOVABS (650,540)
CALL DRWREL (0,20)
CALL MOVABS (550,550)
CALL DRWABS (650,550)
CALL MOVABS (725,600)
CALL DRWABS (725,500)
CALL MOVABS (725,550)
CALL DRWABS (825,550)
CALL MOVABS (715,600)
CALL DRWREL (20,0)
CALL MOVABS (715,575)
CALL DRWREL (20,0)
CALL MOVABS (715,550)
CALL DRWREL (10,0)
CALL MOVABS (715,525)
CALL DRWREL (20,0)
CALL MOVABS (715,500)
CALL DRWREL (20,0)
```

```
CALL MOVABS (750,540)
CALL DRWREL (0,20)
CALL MOVABS (775,540)
CALL DRWREL (0,20)
CALL MOVABS (800,540)
CALL DRWREL (0,20)
CALL MOVABS (825,540)
CALL DRWREL (0,20)
CALL MOVABS (900,550)
CALL DRWABS (1000,550)
CALL MOVABS (900,540)
CALL DRWREL (0,20)
CALL MOVABS (925,540)
CALL DRWREL (0,20)
CALL MOVABS (950,540)
CALL DRWREL (0,20)
CALL MOVABS (975,540)
CALL DRWREL (0,20)
CALL MOVABS (1000,540)
CALL DRWREL (0,20)
CALL MOVABS (75,450)
CALL ANCHO (65)
CALL MOVABS (250,450)
CALL ANCHO (66)
CALL MOVABS (425,450)
CALL ANCHO (67)
CALL MOVABS (600,450)
CALL ANCHO (68)
CALL MOVABS (775,450)
CALL ANCHO (69)
CALL MOVABS (950,450)
CALL ANCHO (70)
RETURN
END
```

```
C                         SUBROUTINE PSYMBL
      SUBROUTINE PSYMBL (LTYPE,LINE,ICHAR)
      LTYPE = 0
      LINE  =  0
      ICHAR = 42
      TYPE 10
      CALL IPOKE('44,'10000 .OR. IPEEK('44))
20    LTYPE = ITTINR()
      IF (LTYPE .EQ. 84) GOTO 60
      IF (LTYPE .EQ. 83) GOTO 60
      IF (LTYPE .EQ. 68) GOTO 60
      IF (LTYPE .EQ. 80) GOTO 60
      IF (LTYPE .EQ. 76) GOTO 30
      IF (LTYPE .EQ. 75) GOTO 50
      GOTO 20
30    CONTINUE
      CALL IPOKE('44,'10000 .XOR. IPEEK('44))
      CALL LINEF
      TYPE 35
      ACCEPT 40, LINE
      IF (LINE .LT. 0 .AND. LINE .GT. 4) LINE=0
      GOTO 65
50    CONTINUE
      CALL LINEF
      TYPE 55
57    ICHAR = ITTINR()
      IF (ICHAR .GT. 32 .AND. ICHAR .LT. 127) GOTO 60
      GOTO 57
C
10    FORMAT (' PLOTTING SYMBOL T)RIANGLE S)QUARE D)IAMOND L)INE',
     1 ' P)OINT K)EYBOARD CHAR',$,1X)
35    FORMAT (//,' ENTER',4X,'0 - FOR A SOLID LINE',/
     1                 10X,'1 - FOR A DOTTED LINE',/,
     2                 10X,'2 - FOR A DASH-DOT LINE',/,
     3                 10X,'3 - FOR A SHORT-DASHED LINE',/,
     4                 10X,'4 - FOR A LONG-DASHED LINE'      )
40    FORMAT (I2)
55    FORMAT (' ENTER ANY KEYBOARD CHARACTER',$,1X)
C
60    CONTINUE
      CALL IPOKE('44,'10000 .XOR. IPEEK('44))
65    CONTINUE
      RETURN
      END

C                         SUBROUTINE TRIANG
      SUBROUTINE TRIANG (X,Y)
      CALL MOVEA (X,Y)
      CALL MOVREL (-6,-6)
      CALL DRWREL (12,0)
      CALL DRWREL (-6,12)
      CALL DRWREL (-6,-12)
```

```
      RETURN
      END
C                        SUBROUTINE SQUARE
      SUBROUTINE SQUARE (X,Y)
      CALL MOVEA (X,Y)
      CALL MOVREL (-6,-6)
      CALL DRWREL (0,12)
      CALL DRWREL (12,0)
      CALL DRWREL (0,-12)
      CALL DRWREL (-12,0)
      RETURN
      END
C                        SUBROUTINE DIAMON
      SUBROUTINE DIAMON (X,Y)
      CALL MOVEA (X,Y)
      CALL MOVREL (-6,-6)
      CALL DRWREL (-12,-12)
      CALL DRWREL (12,-12)
      CALL DRWREL (12,12)
      CALL DRWREL (-12,12)
      RETURN
      END
C                        SUBROUTINE KCHAR
      SUBROUTINE KCHAR(X,Y,ICH)
      CALL MOVEA (X,Y)
      CALL ANCHO (ICH)
      RETURN
      END
```

```
C                        SUBROUTINE MINMAX.FOR
C
C
C THIS SUBROUTINE DETERMINES THE MINIMUM AND MAXIMUM VALUES IN AN ARRAY
C                                   ----------WRITTEN BY T. J. GREEN.
C
C
      SUBROUTINE MINMAX (PTS,NUMBER,TMIN,TMAX)
      DIMENSION PTS(NUMBER)
C PLACE THE FIRST VALUE IN THE ARRAY IN TMIN AND TMAX
      TMIN=PTS(1)
      TMAX=PTS(1)
      DO 20 N=1, NUMBER
C FIND THE LOWEST VALUE IN THE ARRAY AND PLACE IT IN TMIN
      IF (PTS(N) .LT. TMIN) TMIN=PTS(N)
C FIND THE LARGEST VALUE IN THE ARRAY AND PLACE IT IN TMAX
      IF (PTS(N) .GT. TMAX) TMAX=PTS(N)
20    CONTINUE
      RETURN
      END
```

```
C                         SUBROUTINE CURVE.FOR
C
C
C THIS SUBROUTINE PLOTS THE CURVE
C                                    --------WRITTEN BY T. J. GREEN
C
C
        SUBROUTINE CURVE(PTS,NPTS,NUM,NFILE,XST,XINC,XMIN1,XMAX1)
        LOGICAL*1 ITICR, ITICL, ITICU, ITICD, IXTICV, ICON
        COMMON XTIC(13),YTIC(13)
        DIMENSION PTS(NPTS), NUM(NFILE), ITRANS(15), IGRLAB(75)
        DIMENSION LTYPE(10), LINTY(10), ICHAR(10), ITITLE(70)
        DIMENSION XINC(NFILE), XST(NFILE), IXLAB(70), IYLAB(70)
        IXMIN = 300
        IXMAX = 725
        IYMIN = 300
        IYMAX = 700
        FACT = 100000.00
        CALL MINMAX (PTS,NPTS,YMIN1,YMAX1)
        DO 50 ISYM=1,NFILE
53      IF (ITTOUR(26) .NE. 0) GOTO 53
        TYPE 55,ISYM
        CALL PSYMBL(LTYPE(ISYM),LINTY(ISYM),ICHAR(ISYM))
50      CONTINUE
57      IF (ITTOUR(26) .NE. 0) GOTO 57
        TYPE 20
        CALL CHARIN (IXLAB,70,NXLAB)
        TYPE 25
        CALL CHARIN (IYLAB,70,NYLAB)
        TYPE 34
        CALL CHARIN (IGRLAB,70,NGRLAB)
        TYPE 35
        CALL CHARIN (ITITLE,70,NTITLE)
C CHOOSE THE AXIS TYPE
70      CONTINUE
        ITICR = .FALSE.
        ITICL = .FALSE.
        ITICU = .FALSE.
        ITICD = .FALSE.
        IXTICV = .FALSE.
        XMIN = XMIN1
        XMAX = XMAX1
        YMIN = YMIN1
        YMAX = YMAX1
        XMAX=(AINT((XMAX+(XMAX*0.01))*FACT))/FACT
        YMAX=(AINT((YMAX+(YMAX*0.01))*FACT))/FACT
        TXMIN = XMIN
        TYMIN = YMIN
        CALL INITT (120)
        CALL HEADER (IAXIS)
100     CONTINUE
        IF (IAXIS .EQ. 65) GOTO 80
```

```
      IF (IAXIS .EQ. 66) GOTO 120
      IF (IAXIS .EQ. 67) GOTO 160
      IF (IAXIS .EQ. 68) GOTO 240
      IF (IAXIS .EQ. 69) GOTO 200
      IF (IAXIS .EQ. 70) GOTO 280
      CALL INITT(120)
      CALL TWINDO (250,800,250,720)
      CALL DWINDO (XMIN1,XMAX1,YMIN1,YMAX1)
      GOTO 9000
80    CONTINUE
      CALL INITT(120)
      CALL TWINDO (IXMIN,IXMAX,IYMIN,IYMAX)
      CALL DWINDO (XMIN,XMAX,YMIN,YMAX)
      CALL MOVEA (XMIN,YMIN)
      CALL DRAWA (XMIN,YMAX)
      CALL DRAWA (XMAX,YMAX)
      CALL DRAWA (XMAX,YMIN)
      CALL DRAWA (XMIN,YMIN)
      ITICR = .TRUE.
      ITICU = .TRUE.
      GOTO 1000
120   CONTINUE
      CALL INITT (120)
      CALL TWINDO (IXMIN,IXMAX,IYMIN,IYMAX)
      CALL DWINDO (XMIN,XMAX,YMIN,YMAX)
      CALL MOVEA (XMIN,YMAX)
      CALL DRAWA (XMIN,YMIN)
      CALL DRAWA (XMAX,YMIN)
      CALL DRAWA (XMAX,YMAX)
      ITICD = .TRUE.
      ITICL = .TRUE.
      GOTO 1000
160   CONTINUE
      CALL INITT (120)
      CALL TWINDO (IXMIN,IXMAX,IYMIN,IYMAX)
      CALL DWINDO (XMIN,XMAX,YMIN,YMAX)
      CALL MOVEA (XMIN,YMAX)
      CALL DRAWA (XMIN,YMIN)
      CALL DRAWA (XMAX,YMIN)
      ITICD = .TRUE.
      ITICL = .TRUE.
      GOTO 1000
200   CONTINUE
      YTMIN = YMIN1
      YTMAX = YMAX1
      IF (YTMAX .GT. 0.0) YTMIN=(-1)*0.5*YTMAX
      IF (YTMAX .LE. 0.0) YTMAX=(-1)*0.5*YTMIN
      YMIN=(-3)*(YTMAX-YTMIN)/10
      YMAX=7*(YTMAX-YTMIN)/10
      CALL INITT (120)
      CALL TWINDO (IXMIN,IXMAX,IYMIN,IYMAX)
      CALL DWINDO (XMIN,XMAX,YMIN,YMAX)
```

```
      CALL MOVEA (XMIN,YMIN)
      CALL DRAWA (XMIN,YMAX)
      CALL MOVEA (XMIN,0.0)
      CALL DRAWA (XMAX,0.0)
      ITICU = .TRUE.
      ITICD = .TRUE.
      ITICR = .TRUE.
      ITICL = .TRUE.
      TYMIN = 0.0
      GOTO 1000
240   CONTINUE
      IF (XMAX .LE. 0.0) XMAX=(-1)*XMIN
      IF (YMAX .LE. 0.0) YMAX=(-1)*YMIN
      XMIN = (-1)*XMAX
      YMIN = (-1)*YMIN
      CALL INITT (120)
      CALL TWINDO (IXMIN,IXMAX,IYMIN,IYMAX)
      CALL DWINDO (XMIN,XMAX,YMIN,YMAX)
      CALL MOVEA (0.0,YMIN)
      CALL DRAWA (0.0,YMAX)
      CALL MOVEA (XMIN,0.0)
      CALL DRAWA (XMAX,0.0)
      ITICU = .TRUE.
      ITICD = .TRUE.
      ITICR = .TRUE.
      ITICL = .TRUE.
      TXMIN = 0.0
      TYMIN = 0.0
      GOTO 1000
280   CONTINUE
      IF (YMAX .GT. 0.0) YMIN=(-1)*0.5*YMAX
      IF (YMAX .LE. 0.0) YMAX=(-1)*0.5*YMIN
      CALL INITT (120)
      CALL TWINDO(IXMIN,IXMAX,IYMIN,IYMAX)
      CALL DWINDO (XMIN,XMAX,YMIN,YMAX)
      CALL MOVEA (XMIN,0.0)
      CALL DRAWA (XMAX,0.0)
      ITICU = .TRUE.
      ITICD = .TRUE.
      TYMIN = 0.0
      GOTO 1000
1000  CONTINUE
C DETERMINE THE TIC MARK VALUES
      XRANGE = XMAX - XMIN
      YRANGE = YMAX - YMIN
      CALL TICMAR (XRANGE,YRANGE,XTIC,YTIC)
      DO 1005 I=1,13
      XTIC(I) = XTIC(I) + XMIN
      YTIC(I) = YTIC(I) + YMIN
1005  CONTINUE
C DRAW THE X-AXIS TIC MARKS
      IXTICV = .TRUE.
```

```
      DO 1010 IT=1,13
      IF (XTIC(IT) .GT. XMAX) GOTO 1015
      IF (ITICU) CALL XTICU(XTIC(IT),TYMIN)
      IF (ITICD) CALL XTICD(XTIC(IT),TYMIN)
1010  CONTINUE
1015  CONTINUE
C DRAW THE Y-AXIS TIC MARKS
      DO 1100 IT=1,13
      IF (YTIC(IT) .GT..YMAX) GOTO 1105
      IF (ITICR) CALL YTICR(YTIC(IT),TXMIN)
      IF (ITICL) CALL YTICL(YTIC(IT),TXMIN)
1100  CONTINUE
1105  CONTINUE
C DRAW THE CURVE
9000  CONTINUE
      J = 1
      DO 500 IJ=1, NFILE
      NUM1=NUM(IJ)
      XPT = XST(IJ)
      CALL MOVEA (XPT,PTS(J))
      DO 400 I=1, NUM1
      IF (LTYPE(IJ) .EQ. 68) CALL DIAMON (XPT,PTS(J))
      IF (LTYPE(IJ) .EQ. 84) CALL TRIANG(XPT,PTS(J))
      IF (LTYPE(IJ) .EQ. 83) CALL SQUARE(XPT,PTS(J))
      IF (LTYPE(IJ) .EQ. 80) CALL POINTA(XPT,PTS(J))
      IF (LTYPE(IJ) .EQ. 75) CALL KCHAR(XPT,PTS(J),ICHAR(IJ))
      IF (LTYPE(IJ) .EQ. 76) CALL DASHA(XPT,PTS(J),LINTY(IJ))
      XPT = XPT + XINC(IJ)
      J = J + 1
400   CONTINUE
      LDASH = LDASH + 1
500   CONTINUE
      IF (IAXIS .EQ. 10 .OR. IAXIS .EQ. 13) GOTO 660
C PUT IN THE TICMAR VALUES
      ICONTX = 0
      IAD = 0
      IF (XMAX .GE. 100.0) ICONTX=1
      DO 600 IV=1,13
      IF (XTIC(IV+IAD) .GT. XMAX) GOTO 610
      CALL XTICV(XTIC(IV+IAD),TYMIN,ITRANS)
      IAD = IAD + ICONTX
600   CONTINUE
610   CONTINUE
      IF (IAXIS .EQ. 70) GOTO 660
      IAD = 0
      DO 650 IV=1,13
      IF (YTIC(IV+IAD) .GT. YMAX) GOTO 660
      CALL YTICV(YTIC(IV+IAD),TXMIN,ITRANS)
      IAD = IAD + 1
650   CONTINUE
660   CONTINUE
      CALL SEETW (MINX,MAXX,MINY,MAXY)
```

```
      CALL CSIZE (IHORZ,IVERT)
      IXREL=((MAXX-MINX)-(IHORZ*NXLAB))/2
      IYREL=(MAXY-MINY)/2
      IXMINR=0-MINY+5
      IRELY = -30
      IF (IXTICV) IRELY=IRELY+(-35)
      CALL MOVEA (XMIN,YMIN)
      CALL MOVREL (IXREL,IRELY)
      CALL ANSTR (NXLAB,IXLAB)
      CALL MOVEA (XMIN,YMIN)
      CALL MOVREL (IXMINR,(IRELY+(-50)))
      CALL ANSTR (NTITLE,ITITLE)
      CALL MOVABS (5,525)
      CALL ANSTR (NYLAB,IYLAB)
      CALL MOVABS (5,750)
      CALL ANSTR (NGRLAB,IGRLAB)
      CALL MOVABS (5,50)
      CALL ANMODE
      TYPE 690
      CALL IPOKE("44,"10000 .OR. IPEEK("44))
705   ICON = ITTINR()
      IF (ICON .EQ. 67) GOTO 70
      IF (ICON .EQ. 72) GOTO 710
      IF (ICON .EQ. 81) GOTO 710
      IF (ICON .EQ. 82) GOTO 100
      GOTO 705
710   CONTINUE
      CALL IPOKE("44,"10000 .XOR. IPEEK("44))
C*********************** FORMAT STATEMENTS ***********************
20    FORMAT (' ENTER THE X-AXIS LABEL')
25    FORMAT (1X,'ENTER THE Y-AXIS LABEL')
34    FORMAT (1X,'INPUT THE GRAPH LABEL (75',
     1 ' CHARACTERS MAXIMUM)')
35    FORMAT (1X,'INPUT THE TITLE AND/OR COMMENTS (70',
     1 ' CHARACTERS MAXIMUM)')
39    FORMAT (1X,'INPUT THE REMARKS (20 CHARACTERS MAXIMUM)',
     1 ' FOR FILE #',I1)
55    FORMAT (//,' CURVE #',I2)
690   FORMAT (' ENTER  R)EPLOT C)HANGE AXIS H)ARDCOPY OR Q)UIT')
C*****************************************************************
C
      RETURN
      END

      SUBROUTINE XTICU(TM,YMIN)
      CALL MOVEA (TM,YMIN)
      CALL UTIC
      RETURN
      END

      SUBROUTINE XTICD(TM,YMIN)
      CALL MOVEA (TM,YMIN)
```

```
      CALL DTIC
      RETURN
      END

      SUBROUTINE YTICL(TM,XMIN)
      CALL MOVEA (XMIN,TM)
      CALL LTIC
      RETURN
      END

      SUBROUTINE YTICR(TM,XMIN)
      CALL MOVEA (XMIN,TM)
      CALL RTIC
      RETURN
      END

      SUBROUTINE XTICV(TV,YMIN,ITRANS)
      DIMENSION ITRANS(15)
      CALL ASCODE (DBLE(TV),ITRANS,J)
      J=J-4
      CALL MOVEA (TV,YMIN)
      CALL MOVREL ((-3)*J,-35)
      CALL ANSTR (J,ITRANS(1))
      RETURN
      END

      SUBROUTINE YTICV(TV,XMIN,ITRANS)
      DIMENSION ITRANS(15)
      CALL ASCODE (DBLE(TV),ITRANS,J)
      J=J-4
      CALL MOVEA (XMIN,TV)
      CALL MOVREL ((-15)*J-15,0)
      CALL ANSTR (J,ITRANS(1))
      RETURN
      END

      SUBROUTINE CHARIN (IARRAY,IDIM,NCHOUT)
      DIMENSION IARRAY(IDIM)
      CALL LINEF
      NCHOUT = 0
      ICHAR = -1
      DO 30 I=1,IDIM
10    ICHAR = ITTINR()
      IF (ICHAR .GT. 31 .AND. ICHAR .LT. 127) GOTO 20
      IF (ICHAR .EQ. 10) GOTO 40
      GOTO 10
20    CONTINUE
      IARRAY(I) = ICHAR
      NCHOUT = NCHOUT + 1
30    CONTINUE
40    CONTINUE
      RETURN
```

```
C                     SUBROUTINE HEADER.FOR
C
C
C THIS SUBROUTINE DETERMINES THE TYPE OF GRAPH AXIS TO BE USED
C                                 ----------WRITTEN BY T. J. GREEN
C
C
        SUBROUTINE HEADER(IAXIS)
        CALL MOVABS (0,675)
        CALL ANMODE
        TYPE 10
        CALL HEADGR
        CALL MOVABS (0,350)
        CALL ANMODE
        TYPE 20
        CALL IPOKE('44,'10000 .OR. IPEEK('44))
5       IAXIS = ITTINR()
        IF (IAXIS .GE. 65 .AND. IAXIS .LE. 70) GOTO 40
        IF (IAXIS .EQ. 10 .OR.  IAXIS .EQ. 13) GOTO 40
        GOTO 5
40      CONTINUE
C
C*************************** FORMAT STATEMENTS **********************
10      FORMAT (1X,'GRAPH TYPES')
20      FORMAT (1X,'ENTER THE TYPE OF GRAPH AXIS (A,B,C,D,E,F)',$,1X)
C******************************************************************
C
        CALL IPOKE ('44,'10000 .XOR. IPEEK('44))
        RETURN
        END
```

```
C                         SUBROUTINE TICMAR.FOR
C
C
C THIS SUBROUTINE DETERMINES THE X- AND Y-AXIS TICK MARK VALUES
C                                     -----------WRITTEN BY T. J. GREEN
C
C
      SUBROUTINE TICMAR(XRANGE,YRANGE,XTIC,YTIC)
      DIMENSION  XTIC(13), YTIC(13)
      DATA TENTH,TEN,HUND,THOUS/0.1,10.0,100.0,1000.0/
      LCOUNT=13
      THREE=3.0
C FIND THE X-AXIS TIC MARKS
      DIV=10.00
      IF (XRANGE .GT. THREE .AND. XRANGE .LT. TEN) DIV=XRANGE
      IF (XRANGE .LT. 5.0 .OR. XRANGE .GT. 10000.0) DIV=5.0
      XINC=(XRANGE/DIV)
      XTIC(1)=0.0
      DO 9 I=2, LCOUNT
      XTIC(I)=XTIC(I-1)+XINC
9     CONTINUE
C FIND THE Y-AXIS TIC MARKS
      DIV=10.0
      YINC=(YRANGE/DIV)
      YTIC(1)=0.0
      DO 11 I=2, LCOUNT
      YTIC(I)=YTIC(I-1)+YINC
11    CONTINUE
C
      RETURN
      END
```

```
(*$E+*)
   TYPE BLOCK=ARRAY[0..1024]OF CHAR;
        RECODE=ARRAY[0..1]OF INTEGER;
        CONTROL=ARRAY[0..127]OF CHAR;

PROCEDURE SENDM(VAR SOCNUM,DATLEN,CONLEN,DESTH,STATUS:INTEGER;
                DATBUF:BLOCK; CONDAT:CONTROL; RESULT:RECODE);
   VAR CCB:ARRAY[0..5] OF INTEGER;
       COUNT,I:INTEGER;
   BEGIN
     IF (SOCNUM=128) OR (SOCNUM=144) THEN CONLEN:=0;
     COUNT:=1;
     CCB[0]:=64; CCB[1]:=0; CCB[2]:=SOCNUM;
     CCB[3]:=0;   CCB[4]:=0; CCB[5]:=CONLEN;
     RESULT[0]:=255; RESULT[1]:=0;
     REPEAT
(*$C
;
CSR=^0166000
CAP=^0166002
VEC=^0460
;
        NOP
        CLR     %2
        CLR     %3
        CLR     %4
        MOV     #20.,%1
        MOV     %6,%2
        ADD     #RESULT,%2
        MOV     %6,%4
        ADD     #CCB,%4
        SWAB    %2
        MOVB    %2,2(%4)
        SWAB    %2
        MOVB    %2,3(%4)
        MOV     %6,%2
        ADD     #DATBUF,%2
        SWAB    %2
        MOVB    %2,6(%4)
        SWAB    %2
        MOVB    %2,7(%4)
        MOV     @DATLEN(%6),%2
        SWAB    %2
        MOVB    %2,8(%4)
        SWAB    %2
        MOVB    %2,9(%4)
        MOV     @DESTH(%6),11(%4)
        MOV     %4,%2
        MOV     #INT,VEC
AGAIN:  BIS     #^0200,CSR
```

```
         MTPS    %3
LOOP:    TST     CSR
         BPL     LOOP
         MOVB    #0,CAR
LOOP1:   TST     CSR
         BPL     LOOP1
         SWAB    %2
         MOVB    %2,CAR
LOOP2:   TST     CSR
         BPL     LOOP2
         SWAB    %2
         MOVB    %2,CAR
         NOP
WAIT1:   BR      WAIT1
INT:     DEC     %1
         BNE     TRY
         BR      STO
TRY:     CMP     #255.,RESULT(%6)
         BEQ     AGAIN
STO:     MOV     #^O340,%3
         ADD     #4,%6
         MTPS    %3
;
*)
          COUNT:=COUNT + 1;
        UNTIL (RESULT[0] <> 255) OR (COUNT = 10000);
        STATUS:=RESULT[0];
     END;
```

```
(*$E+*)
PROCEDURE INTRAN;
   VAR CCB:ARRAY[0..1] OF INTEGER;
       RESULT,COUNT,I:INTEGER;
   BEGIN
     COUNT:=1;
     CCB[0]:=32;
     RESULT:=255;
     REPEAT
(*$C
;
CSR=^O166000
CAR=^O166002
VEC=^O460
;
         CLR      %3
         MOV      #20.,%1
         MOV      %6,%2
         ADD      #RESULT,%2
         MOV      %2,%0
         MOV      %6,%4
         ADD      #CCB,%4
         SWAB     %2
         MOVB     %2,2(%4)
         SWAB     %2
         MOVB     %2,3(%4)
         MOV      %4,%2
         MOV      #INT,VEC
         MTPS     %3
AGAIN:   NOP
LOOP:    TST      CSR
         BPL      LOOP
         MOVB     #0,CAR
LOOP1:   TST      CSR
         BPL      LOOP1
         SWAB     %2
         MOVB     %2,CAR
LOOP2:   TST      CSR
         BPL      LOOP2
         SWAB     %2
         MOVB     %2,CAR
WAIT1:   BR       WAIT1
INT:     ADD      #4,%6
         DEC      %1
         BNE      TRY
         BR       STO
TRY:     CMPB     #255.,(%0)
         BEQ      AGAIN
STO:     NOP
;
```

```
*)
      COUNT:=COUNT + 1;
     UNTIL (RESULT <> 255) OR (COUNT = 10000);
     IF COUNT >= 10000 THEN
                BEGIN
                FOR I:=1 TO 5 DO
                WRITELN(CHR(7));
                FOR I:=1 TO 8 DO
                WRITE(CHR(7),'FAIL TO INITAILIZE THE TRANSPORTER AFTER ',
                        COUNT:5,' TRIES    ');
                WRITELN; WRITELN; WRITELN;
(**$C
;
        .MCALL  .EXIT
        .EXIT
;
*)
                END;
   END;
```

```
(*$E+*)
    TYPE RECODE=ARRAY[0..1]OF INTEGER;
         CONTROL=ARRAY[0..127]OF CHAR;

PROCEDURE SETUP(VAR SOCNUM,DATLEN,CONLEN,STATUS,DATADR:INTEGER;
                    CONDAT:CONTROL; RESULT:RECODE);
   VAR CCB:ARRAY[0..5] OF INTEGER;
        COUNT,I:INTEGER;
   BEGIN
     IF (SOCNUM=128) OR (SOCNUM=144) THEN CONLEN:=0;
     COUNT:=1;
     CCB[0]:=240; CCB[1]:=0; CCB[2]:=SOCNUM;
     CCB[3]:=0;   CCB[4]:=0; CCB[5]:=CONLEN;
     RESULT[0]:=255; RESULT[1]:=0;
     REPEAT
(*$C
;
CSR=^0166000
CAR=^0166002
VEC=^0460
;
        CLR     %3
        CLR     %4
        MOV     #20.,%1
        MOV     %6,%2
        ADD     #RESULT,%2
        MOV     %6,%4
        ADD     #CCB,%4
        SWAB    %2
        MOVB    %2,2(%4)
        SWAB    %2
        MOVB    %2,3(%4)
        MOV     @DATADR(%6),%2
        SWAB    %2
        MOVB    %2,6(%4)
        SWAB    %2
        MOVB    %2,7(%4)
        MOV     @DATLEN(%6),%2
        SWAB    %2
        MOVB    %2,8(%4)
        SWAB    %2
        MOVB    %2,9(%4)
        MOV     %4,%2
        MOV     #INT,VEC
 AGAIN: BIS     #^0200,CSR
        MTPS    %3
 LOOP:  TST     CSR
        BPL     LOOP
        MOVB    #0,CAR
 LOOP1: TST     CSR
```

```
         BPL      LOOP1
         SWAB     %2
         MOVB     %2,CAR
LOOP2:   TST      CSR
         BPL      LOOP2
         SWAB     %2
         MOVB     %2,CAR
WAIT1:   BR       WAIT1
INT:     DEC      %1
         HALT
         BNE      TRY
         BR       STO
TRY:     CMP      #255.,RESULT(%6)
         BEQ      AGAIN
STO:     MOV      #^O340,%3
         ADD      #4,%6
         MTPS     %3
;
*)
       COUNT:=COUNT + 1;
     UNTIL (RESULT[0] <> 255) OR (COUNT = 10000);
     STATUS:=RESULT[0];
   END;
```

```
(*$E+*)

PROCEDURE ENDREV(VAR SOCNUM,STATUS:INTEGER);
   VAR CCB:ARRAY[0..2] OF INTEGER;
       RESULT,COUNT,I:INTEGER;
   BEGIN
     COUNT:=1;
     CCB[0]:=16; CCB[1]:=0; CCB[2]:=SOCNUM;
     RESULT:=255;
     REPEAT
(*$C
;
CSR=^O166000
CAR=^O166002
VEC=^O460
;
        CLR     %2
        CLR     %3
        CLR     %4
        MOV     #20.,%1
        MOV     %6,%2
        ADD     #RESULT,%2
        MOV     %6,%4
        ADD     #CCB,%4
        SWAB    %2
        MOVB    %2,2(%4)
        SWAB    %2
        MOVB    %2,3(%4)
        MOV     %4,%2
        MOV     #INT,VEC
AGAIN:  BIS     #^O200,CSR
        MTPS    %3
LOOP:   TST     CSR
        BPL     LOOP
        MOVB    #0,CAR
LOOP1:  TST     CSR
        BPL     LOOP1
        SWAB    %2
        MOVB    %2,CAR
LOOP2:  TST     CSR
        BPL     LOOP2
        SWAB    %2
        MOVB    %2,CAR
        NOP
WAIT1:  BR      WAIT1
INT:    DEC     %1
        BNE     TRY
        BR      STO
TRY:    CMP     #255.,RESULT(%6)
        BEQ     AGAIN
```

```
STO:    MOV     #^O340,%3
        ADD     #4,%6
        MTPS    %3
;
*)
      COUNT:=COUNT + 1;
     UNTIL (RESULT <> 255) OR (COUNT = 10000);
     STATUS:=RESULT;
   END;
```

```
PROGRAM OMNET;
(* A PROGRAM TO SET UP THE OMNINET TRANSPORTER COMMANDS *)

  TYPE  BLOCK1 = ARRAY[0..1] OF INTEGER;
        BLOCK2 = ARRAY[0..2] OF INTEGER;
        BLOCK5 = ARRAY[0..5] OF INTEGER;
        BUFF01 = PACKED ARRAY[0..80] OF CHAR;
        BUFF02 = ARRAY[0..127] OF CHAR;

  VAR HOST,CODE,NUM,CLEN,DLEN,SOCNUM:INTEGER;
      BUFDAT:BUFF01;
      REPLY:BOOLEAN; CH:CHAR;

  PROCEDURE OMNB1(VAR CCB2:BLOCK2; VAR RECODE:INTEGER);
  EXTERNAL;

  PROCEDURE OMNB2(VAR CCB5:BLOCK5; DATABUF:BUFF01;
                  CONBUF:BUFF02; VAR RESULT1:BLOCK1;
                  LENGTH,DESTH:INTEGER             );
  EXTERNAL;


  PROCEDURE WHOAMI(VAR RESULT:INTEGER);

    VAR CMB:BLOCK2;

    BEGIN
     RESULT:=255;
     CMB[0]:=01;
     OMNB1(CMB,RESULT);
    END;

  PROCEDURE INTRAN(VAR RESULT:INTEGER);

    VAR CMB:BLOCK2;

    BEGIN
     RESULT:=255;
     CMB[0]:=32;
     OMNB1(CMB,RESULT);
    END;

  PROCEDURE ECHO(VAR DESTH,RESULT:INTEGER);

    VAR CMB:BLOCK2;

    BEGIN
     CMB[0]:=02;
     CMB[2]:=DESTH;
     RESULT:=255;
     OMNB1(CMB,RESULT);
```

```
  END;

PROCEDURE ENDREC(VAR SOCKET,RECODE:INTEGER);

  VAR CMB:BLOCK2;
      RESULT:INTEGER;

  BEGIN
   RESULT:=255;
   CMB[0]:=16; CMB[2]:=SOCKET;
   OMNB1(CMB,RESULT);
   RECODE:=RESULT;
  END;

PROCEDURE SEND(MESSG:BUFF01; SOCKET,DHOST,DATLEN,CLENG:INTEGER;
               VAR RECODE:INTEGER                              );

  VAR CMB5:BLOCK5;
      RESULT1:BLOCK1;
      BUFCON:BUFF02;

  BEGIN
   RESULT1[0]:=255;
   CMB5[0]:=64; CMB5[2]:=SOCKET;
   CMB5[5]:=CLENG;
   OMNB2(CMB5,MESSG,BUFCON,RESULT1,DATLEN,DHOST);
   RECODE:=RESULT1[0];
  END;

PROCEDURE SETUP(MESBUF:BUFF01;SOCKET,DATLEN,CLENG:INTEGER;
                VAR RECODE:INTEGER                        );

  VAR DEST:INTEGER; CMB5:BLOCK5;
      RESULT1:BLOCK1; BUFCON:BUFF02;

  BEGIN
   RESULT1[0]:=255;
   CMB5[0]:=240; CMB5[2]:=SOCKET;
   CMB5[5]:=CLENG; DEST:=0;
   OMNB2(CMB5,MESBUF,BUFCON,RESULT1,DATLEN,DEST);
   RECODE:=RESULT1[0];
  END;


  BEGIN

  END.
```

```
;------------------------------------------------------------
; MACRO POPS A 16 BIT ARGUMENT
;
        .MACRO POP
        PLA
        STA     %1
        PLA
        STA     %1+1
        .ENDM
;------------------------------------------------------------
; MACRO PUSH A 16 BIT ARGUMENT
;
        .MACRO PUSH
        LDA     %1+1
        PHA
        LDA     %1
        PHA
        .ENDM
;-----------------------------------------------------------


        .PROC OMNB1,2   ;TWO PARAMETERS
;-----------------------------------------------------------
; ROUTINE TO SEND 4 BYTES OR LESS
;    TO THE OMNINET TRANSPORTER
;
; PROCEDURE OMNB1(VAR CCB2:BLOCK2; VAR RECODE:INTEGER);
;-----------------------------------------------------------
RETURN  .EQU    0
TEMCCB  .EQU    2
RESADR  .EQU    4
CCBADR  .EQU    6
SCAR    .EQU    0C0E0

        POP     RETURN
        POP     RESADR
        POP     TEMCCB
        LDA     TEMCCB
        STA     CCBADR
        LDA     TEMCCB+1
        STA     CCBADR+1
        LDY     #2
        LDA     RESADR+1
        STA     @CCBADR,Y
        LDY     #3.
        LDA     RESADR
        STA     @CCBADR,Y
LOOP    BIT     SCAR
        BPL     LOOP
        LDA     #0.
        STA     SCAR
```

```
LOOP1   BIT     SCAR
        BPL     LOOP1
        LDA     CCBADR+1
        STA     SCAR
LOOP2   BIT     SCAR
        BPL     LOOP2
        LDA     CCBADR
        STA     SCAR
LOOP3   BIT     SCAR
        BPL     LOOP3
        PUSH    RETURN
        RTS
        .END
```

```
;--------------------------------------------------------
; MACRO POP 16 BIT ARGUMENT
;
        .MACRO POP
        PLA
        STA     %1
        PLA
        STA     %1+1
        .ENDM
;--------------------------------------------------------
; MACRO PUSH 16 BIT ARGUMENT
;
        .MACRO PUSH
        LDA     %1+1
        PHA
        LDA     %1
        PHA
        .ENDM
;--------------------------------------------------------
; MACRO TRANSFER A 16 BIT ARGUMENT
;
        .MACRO TRAN
        LDA     %1
        STA     %2
        LDA     %1+1
        STA     %2+1
        .ENDM
;--------------------------------------------------------


        .PROC OMNB2,5   ;FIVE PARAMETERS
;--------------------------------------------------------
; ROUTINE TO SEND 5 BYTES OR MORE TO
;       THE OMNINET TRANSPORTER
;
; PROCEDURE OMNB2 (VAR CCB5:BLOCK5; DATABUF:BUFF01;
                   CONBUF:BUFF02; VAR RESULT1:BLOCK1;
                   LENGTH,DESTH:INTEGER          );
;--------------------------------------------------------
RETURN  .EQU    0
TEMCCB  .EQU    2
TEMBF1  .EQU    4
TEMBF2  .EQU    6
TEMRES  .EQU    8
LENADR  .EQU    0A
DESADR  .EQU    0C
CCBADR  .EQU    0E
BF1ADR  .EQU    10
RESADR  .EQU    14
SCAR    .EQU    0C0E0
```

```
        POP     RETURN
        POP     DESADR
        POP     LENADR
        POP     TEMRES
        POP     TEMBF1
        POP     TEMCCB
        TRAN    TEMCCB,CCBADR
        TRAN    TEMBF1,BF1ADR
        TRAN    TEMRES,RESADR
        LDY     #2
        LDA     RESADR+1
        STA     @CCBADR,Y
        LDY     #3.
        LDA     RESADR
        STA     @CCBADR,Y
        LDY     #6
        LDA     BF1ADR+1
        STA     @CCBADR,Y
        LDY     #7
        LDA     BF1ADR
        STA     @CCBADR,Y
        LDY     #8
        LDA     LENADR+1
        STA     @CCBADR,Y
        LDY     #9
        LDA     LENADR
        STA     @CCBADR,Y
        LDY     #11.
        LDA     DESADR
        STA     @CCBADR,Y
LOOP    BIT     SCAR
        BPL     LOOP
        LDA     #0.
        STA     SCAR
LOOP1   BIT     SCAR
        BPL     LOOP1
        LDA     CCBADR+1
        STA     SCAR
LOOP2   BIT     SCAR
        BPL     LOOP2
        LDA     CCBADR
        STA     SCAR
LOOP3   BIT     SCAR
        BPL     LOOP3
        PUSH    RETURN
        RTS
        .END
```