CASE WESTERN RESERVE UNIVERSITY

GRADUATE STUDIES

We hereby approve the thesis of

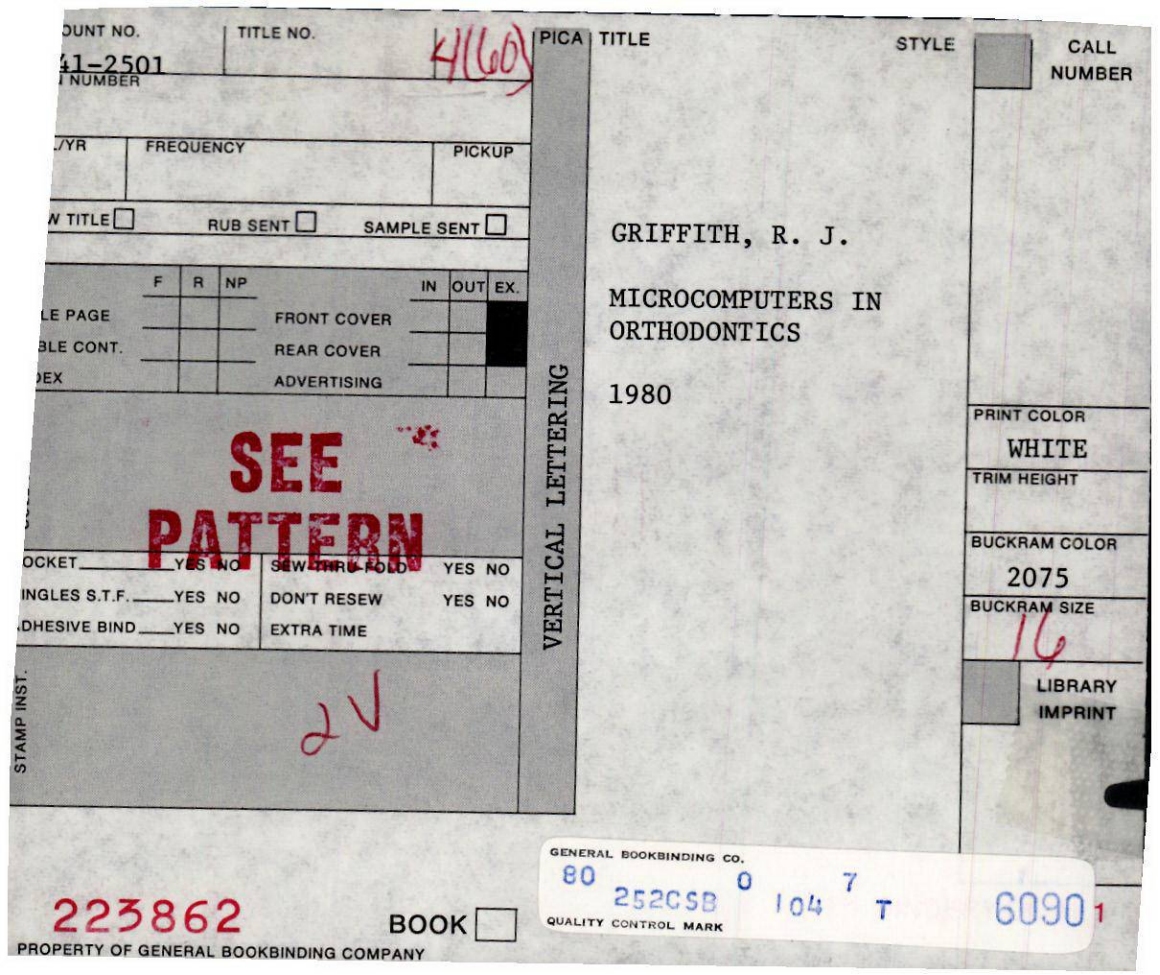
RICHARD J. GRIFFITH

candidate for the MASTER OF SCIENCE IN DENTISTRY degree.

Signed: Ralph J. Bellucci D.D.S.
(Chairman)

Donald H Enlow

Date MAY 23, 1980


OUNT NO.
41-2501
NUMBER
TITLE NO.
41605
PICA
TITLE
STYLE
CALL NUMBER
/YR
FREQUENCY
PICKUP
W TITLE
RUB SENT
SAMPLE SENT
F R NP
IN OUT EX.
LE PAGE
BLE CONT.
DEX
FRONT COVER
REAR COVER
ADVERTISING
SEE PATTERN
OCKET YES NO
INGLES S.T.F. YES NO
DHESIVE BIND YES NO
SEW THRU FOLD YES NO
DON'T RESEW YES NO
EXTRA TIME
VERTICAL LETTERING
STAMP INST.
2V
GRIFFITH, R. J.
MICROCOMPUTERS IN ORTHODONTICS
1980
PRINT COLOR
WHITE
TRIM HEIGHT
BUCKRAM COLOR
2075
BUCKRAM SIZE
16
LIBRARY IMPRINT
GENERAL BOOKBINDING CO.
80 0 7
252CSB 104 T 60901
QUALITY CONTROL MARK
223862
BOOK
PROPERTY OF GENERAL BOOKBINDING COMPANY

MICROCOMPUTERS IN ORTHODONTICS

by

RICHARD J. GRIFFITH

Submitted in partial fulfillment of the requirements

for the Degree of Master of Science in Dentistry

Department of Orthodontics

School of Dentistry

CASE WESTERN RESERVE UNIVERSITY

May, 1980

MICROCOMPUTERS IN ORTHODONTICS

Abstract

by

RICHARD J. GRIFFITH

Computer systems in the form of microcomputers have made electronic information processing an economic feasibility for small organizations. Two organizations which can derive benefits from the microcomputer are a dental office and a department within a dental school.

Microcomputers may be used for business functions. They cal also be used in clinical application or for research purposes. Unfortunately, there are few dentists who have the background to evaluate their needs for a computer system and to then proceed with with procurement and implementation of an appropriate system.

The purposes of this study are as follows:

1. Design a microcomputer hardware configuration for the Orthodontic Department of the Case Western Reserve University Dental School.
2. Select the specific types of hardware and their manufacturers.
3. Select a qualified and competent distributor.
4. Determine applications and their priority for the Department of Orthodontics at the Case Western Reserve University Dental School.

5. Design, write, and implement programs for the selected microcomputer system that could be used for clinical or research purposes.

6. Provide an operations manual that would enable a student, dentist, or auxillary to use the system.

7. Evaluate the entire process and provide pertinent comments and recommendations for others who might be interested in microcomputer systems.

A six component system was configured for the Orthodontic Department. Using this system, nine programs were developed. Seven of those were created for research purposes and two primarily for clinical application. An operations manual was written to facilitate the use of the programs.

In addition, suggested recommendations for dentists interested in microcomputer systems have been set forth.

ACKNOWLEDGEMENTS

The author wishes to extend his sincere appreciation to the following individuals and groups:

Dr. Rolf G. Behrents, Director of the Orthodontic Clinic, Department of Orthodontics, School of Dentistry, Case Western Reserve University, for his assistance as chairman of my research committee.

Dr. Donald Enlow, Assistant Dean for Graduate Studies, Professor and Chairman of the Department of Orthodontics, School of Dentistry, Case Western Reserve University, for his kindness to me as an individual and his dedication to orthodontic education.

Dr. Ike Slodov, for making this project possible by donating a microcomputer system to the Orthodontic Department. I wish to also thank Dr. Slodov for all of the personal interest he has shown me and my class members.

Mr. William Golden, for the excellent graphics on the program flowcharts.

Mrs. Linda Nicholl, for her skills in typing this thesis.

Dr. Donald Gustovich, for bringing levity into many difficult times.

Dr. James Trouten, for his thoughtfulness and friendship.

Dr. William Koenig, my partner and close friend, for arranging these two years for me.

The faculty of the Orthodontic Department, for providing a most outstanding education for me and my classmates.

The staff, for their help in providing an environment that facilitated learning.

DEDICATED

to

Mr. and Mrs. R. Jack Griffith

My parents,
whose love and encouragement
have helped me achieve my education

Dr. and Mrs. Weldon G. Blodgette

My father and mother-in-law,
who have loved me as a son

Pamela B. Griffith

My wife,
whom I love very much

TABLE OF CONTENTS

	Page
ABSTRACT	ii
ACKNOWLEDGEMENTS	iv
DEDICATION	vi
LIST OF ILLUSTRATIONS	viii
INTRODUCTION	1
REVIEW OF THE LITERATURE	2
MATERIALS AND METHODS	15
RESULTS	20
DISCUSSION	60
SUMMARY AND CONCLUSIONS	71
LITERATURE CITED	73
APPENDICES	75

LIST OF ILLUSTRATIONS

Page

1. ANGL3--Point entry example 46
2. ANGL4--Point entry example 50

INTRODUCTION

Two forces are responsible for bringing electronic information processing directly to the dentist and dental researchers. One is the demand on each of these individuals to have large quantities of relative information readily available. The other is the technological progress that has been made in the field of electronics to cause a significant decrease in the costs associated with procuring and operating a computer system. Together, these factors make microcomputer systems a powerful adjunct for a dental practice and for dental research.

The purposes of this study are as follows:

1. Design a microcomputer hardware configuration for the Orthodontic Department of the Case Western Reserve University Dental School.
2. Select the specific types of hardware and their manufacturers.
3. Select a qualified and competent distributor.
4. Determine applications and their priority for the Department of Orthodontics at the Case Western Reserve University Dental School.
5. Design, write, and implement programs for the selected microcomputer system that could be used for clinical or research purposes.
6. Provide an operations manual that would enable a student, dentist, or auxillary to use the system.
7. Evaluate the entire process and provide pertinent comments and recommendations for others who might be interested in microcomputer systems.

LITERATURE REVIEW

Definition

A computer is a mechanism that can accept information; store information; retrieve it either immediately or at some future time; perform operations on information; and present information in a usable format.

Historical

Computers, as they are known today, were not conceived suddenly following World War II. Their history reaches back many hundreds, if not several thousand years. A most effective and popular computational device of early civilization was the abacus. Early navigational instruments may be regarded as simple computers.

As the civilized world moved into the era of the industrial revolution the need for more sophisticated data processing devices increased. The slide rule became an important tool for increasing the speed by which calculations can be performed. In the early 1800's an Englishman, Charles Babbage proposed the "Analytical Engine" (Reid-Green, 1978). He was never able to produce his engine, but the ideas were harbingers of many things to come. Babbage suggested that data be stored as digits represented by holes punched in heavy paper.

In the mid to late 1800's, an American, Herman Hollerith, used his knowledge of data processing and the availability of electrical power to develop a data processing machine (Reid-Green, 1978). This device utilized a punched card system. In 1890, the United States Census Bureau used Hollerith's system for that year's census. The card system was successful in handling the increased volume of material and was capable of performing all necessary functions in about one third of the previously required time.

First Generation Computers

The first half of the Twentieth Century produced many variations of mechanical processing systems. With the development of the vacuum tube, the capability to transmit and store electrical pulses became available. Eventually, this led to the first truly electronic computers of the 1940's. These computers were known as first generation computers.

In 1944, the University of Pennsylvania developed ENIAC, the prototype first generation computer. It was a vacuum tube and wiring board modality computer. Primitive by today's standards, ENIAC was, however, the first real breakthrough from mechanical data processing into electronic data processing.

First generation computers were the products of extensive research and development. They held out to science and industry the possibility for very large processing increases in information volume. These first computers, amazing as they were, were not without some very significant problems.

The space requirements for early computer systems was a function of the vacuum tube. The vacuum tube is a relatively large piece of electronic equipment, and thousands were required to make up a computer. These tubes generate considerably high temperatures. High temperatures necessitate a distance requirement between the tubes, else they would inhibit air circulation and quickly burn each other out. Early computers grew in bulk even more when required cooling equipment was added. Some computer systems even had electronic water cooling systems.

Reliability was one term which could not be applied to first generation computers. The many parts, the new technology, and the high temperatures were causes of frequent breakdowns. Repair of these computers was a complicated process for which few technicians were suitably trained. Consequently, the users of these early machines had a great amount of time and expense invested in maintenance.

Costs and problems in implementation, operations, and maintenance often outweighed the benefits of the early computers. Few institutions could economically justify the expenditures required for a computer.

Second Generation Computers

In 1949, three men were responsible for a landmark scientific discovery. Bardeen, Brattain and Schockley (Orear, 1967) transformed the theory of semiconductors into what was to become

known as the transistor. This was the technological breakthrough necessary to make computers practical and economical.

The transistor had an unprecedented impact on the computer industry as well as society as a whole. Data processing or information processing began to develop into a science. The cost-productivity relationship of electronic data processing dropped into a range that made it practical for many large companies, governmental agencies, and universities to operate second generation computer systems. Reliability, less stringent economical requirements, greater ease of implementation, simpler programming procedures and more sophisticated information storage capabilities made the second generation, transitorized computer very desirable for high volume data processing. Now information could be processed economically and in mere fractions of previously required time.

Third Generation Computers

The transistor represents one of the greatest advances in modern electronics. It became the basis for a multitude of products that led from transistor to integrated circuit. The integrated circuit combined many semiconductor functions and their interconnections into a silicon chip. Electronic circuitry was reduced to a microscopic scale. What had once required many wiring boards and tubes could now be performed on a chip tiny enough to be held between one's fingers. With this, the computer

industry entered the third generation, the generation of the integrated circuit.

Today the computer industry is still within the third generation concept. Integrated circuits are profoundly more sophisticated than they originally were but the basic idea is the same. These small components are responsible for the decrease in size of computers. They have resulted in increased reliability and capability of computer equipment. The cost of production of integrated circuits has also plummeted, making computers less expensive than they have ever been before.

The computer industry has spawned many specialty interests. Among these are the minicomputer and the microcomputer industries. Minicomputers are the next step down from major computer systems. Their spectrum of capabilities is not quite up to the level of the large systems. However, specialized minicomputer systems may be more appropriate for specific functions, and at a cost well below the large general systems.

The microcomputer industry is the newest development in the data processing field. It has evolved from two directions: from the specialized minicomputer, which still requires a relatively large investment for both purchase and operation; and from the hand calculator, which is incapable of advanced-level information collection, storage, and processing.

The microcomputer is a reduced size and capacity general computer. It has been designed primarily for use by small

businesses and laboratories; by hobbyists; and for personal computing needs. It is less sophisticated in its capabilities than the minicomputer but costs are usually below $20,000. The majority of microcomputer systems available today range from between $1,000 to $10,000 in purchase price.

Memory

Internal Memory

Two categories of memory exist, internal memory and external memory. Internal memory is often referred to as the core of the computer. This memory may be read only memory (ROM) or programmable memory (PROM). The ROM memory is hard-wired into the computer and can not be changed by the user. PROM is available for use by soft-ware systems.

The basic unit of memory is the byte. It, in turn is comprised of eight elements which are termed bits. The position of the bits and their condition is important in determining what the byte is expressing. The entire byte may be considered a binary representation of some character. The bits may be either on, hence their value is one, or they are off with a value of zero. Only these two possibilities exist. With eight bits there may be 256 possible combinations of ones and zeros. From these combinations it is possible to express numerical values, letters and characters.

Memory is often referred to in thousands of bytes. The letter K is used to designate thousands. Microcomputers usually have between 8K and 64K.

External Memory

External memory comes in many forms. This type of memory is often referred to as storage memory. Punched cards were one of the first forms of storage memory. On each card are twelve rows and eighty columns. The position of a single punch on a row and column represents a particular number, letter or character. Punched tape is another means of data storage similar to punched cards.

Magnetic forms of data storage are efficient and easy to use. Magnetic tape is one means of storing large quantities of data. On magnetic tape, data is stored serially, i.e., each record is physically stored in the sequence in which it was written. The major limitation is that the tape must be sequentially read to find a specific record.

Magnetic drums and discs are by far the most popular means of storing information. Discs are now capable of storing many millions of bytes of information and still maintain a random access capability. Random access refers to the ability to access a record directly from a file without having to search an entire file to find that record. This capability is an absolute necessity for those systems that require immediate access to specific records. For microcomputer systems, floppy disc storage or hard disc storage is available. The hard discs generally have a greater storage capacity and shorter data access time.

Another type of storage is becoming available. It is referred to as "bubble memory". The memory actually stores information by producing bubbles with magnetic fields. This type of storage exists in an integrated circuit making it faster than other forms of magnetic storage. It is also non-volatile, which means it is not lost when the power is removed, as is the case with internal programmable memory.

Hardware

Hardware is the term used to describe the physical elements of the computer system. Hardware ranges from microcircuits to the nuts and bolts that hold the frames together. Sometimes the phrase peripheral hardware may be used. This implies that a piece of hardware is not part of the central processing unit. Examples of peripheral hardware would be disc drives, printers, plotters, tape drives, and cathode-ray tube terminals.

A further discussion of the hardware used for the development of this project may be found in the Materials and Methods section of this paper.

Software

Software is the least understood aspect of computer systems for individuals who have had little or no experience with computers. The software is everything that is left over after the hardware and energy are removed. It is the logic that controls the hardware for

some meaningful purpose.

Operating System

The systems software can be divided into two categories. The operating system is the major controlling software of the computer system. It provides the system with a means of using memory, peripheral devices, communicating with the operator, and for using application programs.

Utility Software

The utility software is the other aspect of the systems software. The utility programs provide the user with many capabilities that would otherwise have to be specially written by a programmer. This type of software is very beneficial in the development of user programs. The utilities used in this project are discussed in the Methods and Materials section.

Programming Languages

The software is the logic, but the logic is produced via program languages. When one speaks of low level languages and higher level languages the reference is to how closely the language resembles or approximates machine language. Usually, the more complicated a language the closer it is related to machine language. The higher level languages are more easily understood, require less knowledge of what is actually happening with the computer system, and may be more readily implemented.

Machine language is a binary duplication of the status of the bits in memory. Writing programs in this language gives maximum flexibility and conserves on the memory requirement. Conversely, it is extremely difficult and time consuming to use. This type of programming should be left to the experts.

Assembler language is at the next language level. It is more easily used than machine language but still requires considerable expertise in its usage. Assembler is the language that is used by system software developers where it is necessary to maximize programming efficiency.

The most popular languages are the high level languages. These languages more readily resemble words, phrases and terminology that human beings are accustomed to dealing with. They are relatively easily learned and can be used without intimate knowledge of computer intricasies. COBOL, BASIC, FORTRAN, and PASCAL are examples of high level, structured languages. It is in these languages that most application programs are written. Of course, versatility and efficiency are sacrificed with complexity. This makes it necessary to have several different languages that have been developed for specific types of functions.

Application Software Development

The neophyte attempting to expand into electronic data processing usually looks at only the hardware and pays little attention to software. The great misconception is that all that need

be done is to purchase the hardware, plug it in and information processing begins. Actually, selecting the right equipment probably accounts for only a relatively small part of the work required to get a system running. The remaining effort can be very time consuming and costly. The purchaser may wish to develop his own programs. If he has limited previous experience and cannot work exclusively on the project, much time will be required to learn how to use the equipment, the operating system and the programming languages. Once this has been accomplished it will be necessary to critically analyze the functions of the proposed applications. Then the system must be written, debugged and tested. Many hundreds of man-hours may then be required to implement even a simple system. Several years may be necessary to get more complicated systems into operation.

The cost of development is directly related to the time required for implementation. For the individual doing his own development there are two cost factors. One is the materials and training necessary to get started. The other is the cost of the time the individual spends learning and developing the software.

If the decision is made to hire a systems analyst to design and implement application programs, the time required will be less. However, cost will be significant. An analyst, depending on background and experience, can cost in the range of $10.00 to $25.00 per hour. The cash outlay for the system can easily double.

Expectations

Almost limitless potential exists in the area of electronics. Every indication suggests that the contemporary era of electronics has really just begun. Discoveries and innovations are occurring almost on a daily basis. There is hardly time to put these new ideas into production before something else comes along to make them obsolete. The electronic industry is essentially involved in a technological chain reaction.

The effects on the computer industry are profound. Integrated circuitry is becoming more sophisticated and powerful. The cost of producing electronic elements has decreased. Consequently, the computer industry is an economically strong industry. This alone will stimulate even more research and development. It should require no more than three to five years to make the microcomputers of today seem like tube radios by comparison.

Presently, the microcomputer industry appears to be directing energy in two directions. One is in the development of reliable, cost-effective mega-byte storage (mega refers to million). Hard discs should be the more immediately available mode for storage. Bubble memory is just beginning to be available and its potential is literally open to imagination.

The other direction for development will be the one area that has lagged behind the technological development. This is in the area of software. Good software is very difficult to procure.

That which operates the microcomputer systems is substantially incomplete, especially in comparison to minicomputer systems. Once high quality software packages become available, the cost and difficulty of implementing a system will decrease considerably.

METHODS AND MATERIALS

A computer system may be classified into two major divisions. The one most familiar is the hardware division. The term hardware, as previously mentioned, refers to the physical units or components of the system. Examples are the disc drives, tape drives, printers, and control consoles. The hardwiring, or physical logic in the components, may also be considered part of the hardware.

The second division is the software division. The software is not something that is tangible. It is the logic that controls the complex operation and interactions of the system for useful purposes. This division may be subdivided into two general categories, operating system software and user software.

Hardware System

The computer system hardware used in this project is composed of six elements. The central processing unit has been designed by the Alpha Micro Corporation of Irvin, California. This 16 bit memory processor is responsible for controlling the operation of the system. Floppy disc controllers, device driver, and 48 thousand bytes of main memory are integral parts of the processor. The operating system software and user software become electromagnetically recorded into the internal memory of the central processor when the system is operational.

The primary data storage device is in a dual floppy disc drive designed by the Byte Shop of Ohio in Rocky River, Ohio. This unit has two drives, each capable of using an 8½ inch disc that can store approximately 200,000 bytes of information. The operating system, user programs, and data files are stored on these discs. Because disc drives are used, as opposed to tape storage drives, the system can access information more quickly and can do so in a serial or random access mode.

For hard copy, printed output a Texas Instruments 810 printer designed by Texas Instruments Inc. of Houston, Texas, has been used. The 810 printer is a receive-only printer, i.e., there is no keyboard for communication with the processor. It is a 9 x 7 dot matrix printer with 10 characters per inch and 132 characters maximum per line. The paper that can be handled ranges from 7.6 to 38.1 cm. in width and can have up to one original and five copies.

The console terminal used for controlling and communicating with the operating system and program is a Soroc video terminal, from Soroc Technology Corporation of Anaheim, California. This terminal has a keyboard similar to most typewriter keyboards but with a few additions to facilitate usage.

A Summographics Bit Pack One digitizing board from the Summographics Corporation of Fairfield, Connecticut, was used in the development of programs that store cephalometric points. The digitizer has the capability of resolving two hundred points per

inch. Selection of points is performed with a pen type stylus. The digitizer does not have the continuous line feed option but works in the single point transmission mode. Points selected are transferred into coordinate values and transmitted to a memory buffer in the processor that is accessible by user programs.

The system also has a Tektronix 4006-1 graphics display terminal from Tektronix Inc. of Beamerton, Oregon. This terminal can function in both graphics and alpha modes. Graphics capabilities allow for excellent resolution with 798,720 displayable points out of a total of 1,048,576 addressable points.

Software System

In addition to the aforementioned hardware, the Alpha Micro Operating System (AMOS) provides the software logic for operation of the processor and its peripherals. Within the logic are instruction sequences to transmit, print and read commands. The system provides memory logic for storing and manipulating information in the memory core.

Several adjunct software utilities were used in developing the programs for this project. Most important is the Alphabasic program. All programs that were developed were written in Alphabasic which is an extension of the Basic language. A unique variable description command, called a MAP command, was utilized to design the file records and record fields. Alphabasic also supports triganometric commands which were extensively used in program development.

There was also significant usage of other utilities in the development process. The Copy utility reproduced programs and files. Vue, the text editor software was the utility used for building all of the program source files. The Alphabasic compiler then compiled the source files into machine language object files. The Load program made it possible to link to assembler subroutines in memory. Log provided for operator movement into different sectors. Run loaded programs into memory and then commenced the execution of the program. Sysact initialized discs and established the sectors that were to be used on a disc. Dir displayed the files stored in a sector. (These utilities may be found in the Alphamicro Operating Systems Manual.)

Software support can be found in other programs and program systems that are available from the Alphamicro Corporation. Pascal and Assembler are other available programming languages. Business application systems such as inventory, payroll, accounts payable, accounts receivable, and general ledger may also be obtained.

Methods

A determination was made to develop a program or series of programs that could build a file of cephalometric data points and perform angle and distance calculations among these points. First it was necessary to learn the capabilities of the system software and hardware. This was accomplished through the use of manuals and by writing test programs.

The second step was to design the system of programs that could accomplish the desired goals. The logic was flowcharted from beginning input through file creation and data manipulation.

After the programs were flowcharted, the logic was coded using the Alphabasic language. The coding is first done on paper; then it is transferred to a Basic source file on a disc. Once all of the coding has been transferred to the disc, the program is compiled. Compilations perform several services. It edits the program instructions to determine what syntax errors have been made. These errors are then displayed on the video console. Its other function is to transform the source coding into object coding and to store this in an executable format.

Once all of the errors had been corrected in the coding, it was necessary to test the logic. To do this, dummy files were created. The programs were then run against these files until it was determined that all possible conditions could be accounted for.

After testing was completed, discs were prepared by a support program and then loaded with the operable program. These discs are now available for individuals wishing to do research or clinical analysis.

RESULTS

This manual is prepared to assist users in the operation of the Department of Orthodontics microcomputer system. The manual is divided into two major sections. The first section deals with starting up the system and interactions between the operator and the operating system. The second section gives a description of programs presently available and instructions in their use.

Several precautions should be mentioned before continuing with a discussion of the computer system. When operating the equipment the room temperature should be between the extremes of 65 and 75 degrees farenheit. Beyond these extremes, the probability of system malfunction increases.

Ideally, there should be no smoking in the area of the computer. Smoke or ashes can permanently destroy the sensitive electromagnetic components. This is especially true of the discs upon which programs and data are stored. Deposition of smoke on the disc surface will make them unreadable; consequently all information in the affected file will be lost. Any fluids spilled onto the equipment or the disc will render the device or disc unusable.

When handling the discs, only the paper covered portion should be touched by the operators hands. Touching the delicate

recording surface will leave oil deposits on the surface or will scratch the surface. Either can damage a part of the disc beyond use, thus effectively destroying the entire disc. Always store the disc in a cool, dry area. They should be in individual sheaths at all times except when being used in the disc drive. Do not store the discs in a manner that can result in the abrasion of recording surfaces.

Starting the Equipment

The microcomputer system is composed of six major components and floppy discs upon which the operating system, system utility programs, user programs and user data files are stored.

The central processing unit is the primary controlling device in the entire system. It contains the hardware logic for memory functions and for communication with the other devices. There are two switches on the face of the unit, an on-off switch and a reset switch. Turn the on-off switch to the on position. The reset switch will be used to "boot" the system, which will be explained later. The central processing unit should be the first to be turned on.

There is no particular order that the remaining devices should be turned on. However, it is important to have all devices that are going to be used turned on prior to the booting of the system.

The disc drive unit is a dual floppy disc drive. It has two

slots for mounting the floppy discs. The slot on the left is designated DSKØ and the one on the right is DSK1. DSKØ is the system drive and will contain the system disc. DSK1 is the users drive and holds the users disc which contains the programs and the storage area for data files. There is an on-off switch on the front panel and two spring buttons one next to each disc driver. The spring buttons are used to disengage a disc from the driver. Push the on-off button to the on position. The disc unit should always be turned on after the processor is started. Never insert the discs prior to starting the unit because damage to the recording surface could occur.

The Soroc operator's terminal is the display console through which the operator and the operating system communicate. It is also the device used for interaction with programs. The on-off switch is located on the rear panel. Turn it on at this time. Use of the keyboard will be explained later.

The Texas Instruments printer has an on-off toggle switch on the rear panel. Turn it on. The front control panel has several buttons. The one labeled "on line" should be pressed so that the red indicator light comes on. The "top of form" button will run the paper through a complete cycle. The form alignment buttons can be used to move the forms one print line in either direction. For instructions on how to load and align the paper, refer to the Texas Instruments manual for the model 810 printer.

These four devices, the dual disc drive, the processor, the operator's terminal, and the primter, must be on and operable for the computer system to be used. The other two devices, the digitizer and the graphics terminal are optional for normal operating. Since there are no programs that presently utilize the graphics terminal, its operation will not be discussed in this procedures manual.

The digitizer is activated by a toggle switch on a control box that is attached to the digitizing board via a cable. If there is any intention to use the digitizing board to build data files, then this unit must be turned on prior to the "booting" process.

A thin cable attaches a stylus to the digitizing board. It is by pressing the tip of this stylus upon the board that coordinate values are transmitted to a memory area in the control processing unit. Only press this stylus upon the active area of the board when it is desired to record a point, else erroneous values will be stored in a memory buffer and will then be entered as if they were legitimate data.

Once the equipment has been activated, the discs may be inserted into the disc drive slots. The discs need to be inserted with the slot of the exposed recording surface entered first. The label on the disc should face left relative to the front panel. Insert the disc labeled "System Disc" into DSKØ, the left drive slot, and the disc labeled as the users disc into DSK1, the right drive slot. These discs should be loaded carefully. They must be

inserted all of the way into the drive slot. Remember, only touch the protected part of the disc.

It may be necessary to remove one of the discs. To do so, the disc drive unit must be operating. Pressing the eject button next to the driver will cause the disc to be partially ejected. It may then be removed by hand.

At this time, both discs should be mounted in their respective disc drive slots. The system may now be "booted". Booting refers to the retrieval of the operating system from the systems disc and loading it into memory. To "boot" the system press the "reset" button on the central processing unit. An audible click will indicate that the booting process has started. Verify that the process is taking place by observing the display on the Soroc video terminal. In addition to the video display there should be a small printout (refer to Appendix X). When an electronically created bell sound occurs the system is ready.

Communicating with the Operating System

All communications with the operating system will be performed via the Soroc video terminal. As the system is "booted", information related to the process is displayed on the video screen. As the screen is filled with information, a scrolling process occurs, the first information displayed appears to roll out through the top of the screen. Space is made, then, for an additional line at the base of the screen. When the booting process is complete the system will display a prompt sign and position the cursor for input.

At this time the system is prepared for interaction with the operator.

The prompt sign for the Alpha Micro operating system is a period. When the system is ready to receive input, a period is displayed at the beginning of a print line.

Next to the prompt indicator is the cursor. The cursor is a square white display covering the space of the screen allocated for a character, letter or number. At the position of the cursor a character will appear when a key is depressed.

The Soroc keyboard works similarly to a typewriter keyboard. There are, however, additional keys that are important in the operation of the system. The button marked ALPHA should be in the depressed position. This ensures the transmission of upper case characters. The button labelled RUB is the erase key. By pressing this key the cursor will move to the left one character. It will cause the character it overlaps to be erased. When the operator has completed the entry on the video screen, transmission back to the operating system or to the program in process is accomplished by pressing the RETURN button. All responses that were typed into the console are then transmitted to the processor.

Holding a key depressed will cause multiple printing of the character. The erase function also operates in this manner. Unfortunately, the process of multiple operation occurs if the RETURN key is held down. This can be insignificant if no operations are in progress, however, if a program is running it

may cause the program to operate incorrectly.

System Utility Functions

A system utility, as the name implies, is something that is useful to the operator and can make programming and operations less complicated and more effective. It will be necessary to perform some of the utilities in order for the user to operate the system. In the following explanation the correct entrys for different utility functions will be in bold characters. Because syntax, i.e., commas, colons, letters, etc., is extremely important all console responses will be preceded and followed in this manual by brackets. When entering the clause into the terminal, type in absolutely and only what is between the brackets in this manual. Do not include the brackets.

MOUNT Utility

Sometimes it is necessary to remove a disc from one of the drives, and to replace it with a different disc. After the new disc has been inserted check the video screen. The system prompt, the period, should be to the left side of the screen and the cursor next to it. If this is not so, press the RETURN key and wait for the prompt and cursor to position themselves. When the operator wishes to mount a new disc on the systems drive, the correct entry is [MOUNT DSKØ:]. The correct entry for the user drive would be [MOUNT DSK1:]. The colon is absolutely necessary.

The MOUNT command must also be used if the system is booted without a disc in the user drive and the operator wishes to

mount a disc in that drive.

Press RETURN after the entry is made.

LOG Utility

When the system is booted it is automatically operating from the system disc, DSKØ, and from a specific area, called a partition, in that disc. The partition is given a numerical name, 1,4. The partition 1,4 contains most of the system utilities.

The operator will usually not run programs from partition 1,4 on the system disc but will operate from partition 1,4 on the user disc, DSK1. The user discs are presently set up so that all user programs and data space is on 1,4 of a user disc. To get from 1,4 on DSKØ to 1,4 on DSK1, type in [LOG DSK1:]. To transfer in the opposite direction type in [LOG DSKØ:].

Press RETURN after the entry is made.

The system will display which disc drive and which partition the operator is running on.

DIR Utility

The DIR utility provides directory information to the user. It enables the operator to look at a list of all of the programs that exist in the sector in which the operator is working. Type in [DIR/W].

Press RETURN after the entry is made.

A list of all of the programs and data files will be displayed.

LOAD Utility

This utility allows the operator to bring into memory from a disc a program or subroutine. What is unique with this utility is that the program which is loaded is not executed as it would be if the RUN utility were used. Once the program file is in memory it may then be linked to and used by other programs that are running. The "PLOT" subroutine is an example of a program that will be loaded into memory when a digitized file is being created. Type in [LOAD PLOT.SBR].

Press RETURN after the entry is made.

The system will respond with the system prompt and cursor positioned on the next line. There is no verification that the subroutine has been loaded.

DEL Utility

To clear a program or subroutine from memory the DEL function should be used. This utility is used under present circumstances to delete the "PLOT" subroutine subsequent to building digitized files. The correct entry is [DEL PLOT.SBR].

Press RETURN after the entry is made.

ERASE Utility

This enables the user to erase programs and files from a disc. It essentially makes available space previously allocated to a

file and eliminates that file name from the directory of the disc involved.

The operator will be primarily interested in erasing data files that are no longer needed. There may, however, be some circumstances where it is necessary to erase a program file.

Be sure that the system is logged onto the user disc drive, DSK1.

All of the disc files listed in the directory posess a suffix that identifies the type of file. For data files it is DAT; for program files the suffix may be RUN or BAS. Subroutines have SBR as their suffix.

These suffixes must be included when using the ERASE function. The file name and suffix are separated by a period. Three examples of ERASE commands follow: [ERASE CEFILE.DAT]; [ERASE PLTOD.RUN]; [ERASE PLOT.SBR]. Of course, data file names are arbitrary and named at the discretion of the individual who has created the file.

Press RETURN after the entry is made.

The system will respond by displaying that the file has been deleted and will calculate the amount of new space available for the user.

SYSACT Utility

When using this utility the operator must be extremely careful. Unknowingly, one could completely destroy all the information on the systems disc or the users disc. This utility

erases all files.

This utility is used to initialize a disc. Initialize refers to the software preparation of a disc so that data files and program files may be stored on it. All new discs are initialized before they can be used. Sometimes the directory on a disc will be destroyed. When this happens it is not possible for the system to find programs or data files. To make the disc usable again it should be initialized.

The SYSACT utility also provides for the establishment of partitions on the disc. It was previously mentioned that all user functions will be performed under partition 1,4 so to create any other partitions under the present system will not be necessary.

Do not set pass words for the sectors at this time.

In the example that follows, it will be assumed that the systems disc is mounted.

If not already logged onto DSKØ, do so at this time:

[LOG DSKØ]:

then press RETURN.

Next, the operator will log into a partition other than 1,4. This partition is 1,2. Type in:

[LOG 1,2]

then press RETURN.

The operator will now indicate the utility name to be run on the device on which the disc is to be initialized. Type in:

[SYSACT DSK1]:

and press RETURN.

The utility will return with its own prompt sign, an asterisk (*). Type in an [I]. Then press RETURN. The operator will be asked to confirm that the disc is to be initialized. Answer [Y]
and press RETURN. The asterisks should return. Now type in [A 1,4] and press RETURN. The operator will be asked if a password is desired. Since none is, no answer is to be supplied. Simply press the RETURN key. The asterisk will return.

At this time, the operator can still get out of the utility if there is any doubt about initializing the particular disc. This may be accomplished by holding down the key labeled CONTROL and depressing the C character key at the same time. The system will return with the system prompt. The operator will still be logged into partition 1,2. To get back to 1,4 type in [LOG 1,4] and press RETURN.

If everything is in order with the SYSACT utility the operator can type in an [E] next to the asterisk. Press RETURN and exit from the utility having initialized the disc and established the partition 1,4.

Now type in [LOG 1,4]. Press RETURN.

A new disc is available now on which files may be built.

COPY Utility

This utility is used to reproduce files and to transfer files to a new disc.

To copy the entire systems disc, first log into partition 1,2

of the system disc. Then type in [COPY DSK1:[] = DSKØ:[]]. Press RETURN. The system will copy all of the sectors and all of the files in each partition from the old system disc on DSKØ to the new system disc on DSK1. Each time a file is copied it will be listed on the video screen. Copying an entire disc requires five to ten minutes.

With this command, do not include the outside brackets but do include the bracket pairs following DSK1 and DSKØ.

After the copying process is completed, log back into 1,4. The disc on DSK1 will be an exact copy of the systems disc.

Should it become necessary to recreate a users disc, the process is slightly more complicated. The system disc has very little free space for the transfer of files, consequently the process of copying all the user program files from a back-up disc to a new disc requires several series of COPY, ERASE, and MOUNT functions. The list of necessary instructions follows. Of course, after each entry, the RETURN key is to be pressed.

Mount the disc marked as the program back-up disc on DSK1.

[MOUNT DSK1:]

[COPY DSKØ:=DSK1:PLOT.SBR]

[COPY DSKØ:=DSK1:PLTLOD.BAS]

[COPY DSKØ:=DSK1:SIMDIS.BAS]

Remove the back-up disc and mount on DSK1 the new users disc that has already been initialized.

[MOUNT DSK1:]

[COPY DSK1:=DSKØ:PLOT.SBR]

[COPY DSK1:=DSKØ:PLTLOD.BAS]

[COPY DSK1:=DSKØ:SIMDIS.BAS]

[ERASE PLOT.SBR]

[ERASE PLTLOD.BAS]

[ERASE SIMDIS.BAS]

Remove the new disc and remount the back-up disc on DSK1.

[MOUNT DSK1:]

[COPY DSKØ:=DSK1:ANGL3.BAS]

[COPY DSKØ:=DSK1:ANGL4.BAS]

[COPY DSKØ:=DSK1:PLTEN.BAS]

Remove the back-up disc and mount on DSK1 the new users disc.

[MOUNT DSK1:]

[COPY DSK1:=DSKØ:ANGL3.BAS]

[COPY DSK1:=DSKØ:ANGL4.BAS]

[COPY DSK1:=DSKØ:PILIN.BAS]

[ERASE ANGL3.BAS]

[ERASE ANGL4.BAS]

[ERASE PILIN.BAS]

Remove the new disc and remount the back-up disc on DSK1.

[MOUNT DSK1:]

[COPY DSKØ:=DSK1:ADDIST.BAS]

[COPY DSKØ:=DSK1:MULDST.BAS]

Remove the back-up disc and mount on DSK1 the new users disc.

[MOUNT DSK1:]

[COPY DSK1:=DSKØ:ADDIST.BAS]

[COPY DSK1:=DSKØ:MULDST.BAS]

[ERASE ADDIST.BAS]

[ERASE MULDST.BAS]

Remove the new disc and remount the back-up disc on DSK1.

[MOUNT DSK1:]

[COPY DSKØ:=DSK1:CANLOD.BAS]

Remove the back-up disc and mount on DSK1 the new users disc.

[MOUNT DSK1:]

[COPY DSK1:=DSKØ:CANLOD.BAS]

[ERASE CANLOD.BAS]

Remove the new disc and remount the back-up disc on DSK1.

[MOUNT DSK1:]

[COPY DSKØ:=DSK1:CANNAL.BAS]

Remove the back-up disc and mount on DSK1 the new users disc.

[MOUNT DSK1:]

[COPY DSK1:=DSKØ:CANNAL.BAS]

[ERASE CANNAL.BAS]

At this point a users disc has been built. The programs, are however, in what is called source format. The computer is only capable of using object code. The manner of producing this code will be explained in the next section on compiling programs.

COMPIL Utility

Use this utility after creating a new user disc. It must be

run on each of the programs copied with the exception of "PLOT.SBR". Essentially compile all programs with the suffix BAS. The operator must log onto DSK1 and should be in partition 1,4. While each compilation is being done the system will print out information related to the compilation. Start each compilation only when the system prompt has been displayed in the screen. Press RETURN after each entry.

Below is listed the necessary instructions for compilation of all of the source programs.

[LOG DSK1:]

[COMPIL PLTLOD.BAS]

[COMPIL SIMDIS.BAS]

[COMPIL ANGL3.BAS]

[COMPIL ANGL4.BAS]

[COMPIL PTLIN.BAS]

[COMPIL ADDIST.BAS]

[COMPIL MULDST.BAS]

[COMPIL CANLOD.BAS]

[COMPIL CANNAL.BAS]

If a DIR/W is performed on DSK1 the operator will see that for each program compiled there will be a duplicate program, except it will have the suffix RUN. These programs are now ready to be used.

RUN Utility

When the RUN command is initiated it loads a specified

program into memory and prepares it for execution. A program must have the RUN suffix before it can be executed by the run command. The correct format is [run program name]. Program name is a variable that contains the name found on the disc. The suffix RUN is dropped when using this utility. Following are some examples of how this utility is used.

[RUN PLTLOD]

Press RETURN.

[RUN CANNAL]

Press RETURN.

User Program Operation

There are presently nine programs that may be used by the Case Western Reserve University Department of Orthodontics computer. For research purposes, there is a digitizing program to build a file using X and Y coordinate values taken from cephalometric tracings, and six programs which can be used for cephalometric calculations. Clinical cephalometric calculations are available through the use of two other programs, one to build the disc file and one to perform the predetermined calculations.

In this section of the procedures manual, each program will be discussed thoroughly, and complete instructions will be given on how to run the programs. The operator is instructed to press RETURN after each entry for transmission back to the program or operating system.

Once the operator is running a program, there can be no communication with the operating system. Consequently, the operator should make sure that all discs are inserted, that programs are loaded, and that there is sufficient paper in the printer. An interruption will cause incomplete termination of the program. The problem will then have to be corrected and the program restarted.

Prior to using one of the digitizing programs the operator must be completely organized with respect to the cephalometric material. Tracings should be complete with all points identified that are to be entered from the digitizer. Each cephalogram or tracing must be designated by a six digit identifying number. A list of the points and their sequence of entry should be prepared. Each cephalogram will have recorded the same number of points in the same sequence. If a point cannot be located, either an arbitrary dummy point must be entered or the cephalogram eliminated as an input record.

The list identifying points recorded and their sequence numbers should be used when running the calculation programs. For the calculations, points are identified by sequence number, not by names.

PLTLOD

The "PLTLOD" program is primarily a research tool. It is a program that will give coordinate values to cephalometric points

and store these on a disc file. A maximum of one hundred points may be entered and at least two points must be entered. There is no particular order in which the points are to be entered, but the points have to be entered in the same sequence for each cephalogram digitized, and there must be the same number of points for each cephalogram. A maximum of 90 cephalograms may be loaded on a disc that has no other data files on it. It is the responsibility of the operator to know which points were entered and in what sequence. This is very important when the file is used as an input file for the calculation programs. Records or cephalograms are identified by a six digit identification number.

Following will be the instructions and responses necessary to run this program. Make sure that the system has been booted with the digitizer turned on.

[MOUNT DSK1:]

[LOG DSK1:]

[LOAD PLOT.SBR]

[RUN PLTLOD]

Video display sequence of program prompts:

1. The first video display is a brief program explanation.
2. The second display is an instruction on how to enter the name of a disc file. Another example would be:

   DSK1:TESTI.DAT
3. Next enter the number that is on the label of the disc. This is so the disc may be identified for future use.

4. For the next response, enter the number of points that are to be digitized on each cephalogram.
5. Place the cephalogram on the digitizer and enter a six digit identification number.
6. This display explains the function of the recording surface on the digitizer. Points can only be recorded from within the inner boundaries. Press the stylus all the way down then release it. Do not try to enter the points too quickly. The program will keep track of how many points have been entered.

The program possesses error recovery options at this point. If the wrong point has been entered or the operator forgets which point is being entered, press the stylus to the right of the cephalogram within the right side boundary lines. The last entry will be erased, and displayed upon the screen will be the sequence number of the next point to be entered.

Above the cephalogram, within the upper boundary region is an area for exiting from the program and for error recovery. By pressing the stylus in this region several options will become available.

The first option is to exit from the program. By answering yes, [Y] all files will be closed and the operator will exit from the program.

If the answer is no, [N], the next question will ask if the operator wishes to reset the program. If the answer is yes, [Y],

the operator must restart the program at the point of entering the points to be recorded per cephalogram. All previously recorded records are lost.

If the answer to the reset question is no, [N], the next question will ask if the operator wishes to restart the cephalogram that is wanted. If the answer is yes, [Y], all previously entered points for that record are erased and the operator must restart the cephalogram.

Once a cephalogram has been digitized and recorded on the disc file, it cannot be erased from the file.

7. After the cephalogram has been recorded, the program will ask if the operator wishes to enter another cephalogram. If the answer is no, [N], the files will be closed and the program will provide a print out with the file name, disc number, and the number of cephalograms transferred. If the answer is yes, [Y], the program will ask the operator to mount the next cephalogram and continue with the program as previously done.

To delete the "PLOT" subroutine from memory, type in:

[DEL PLOT.SBR].

SIMDIS

This program calculates the distance between any two points from a digitized cephalogram. Up to fifty two-point distance calculations may be performed per cephalogram. In addition to

calculating the distance between two points, this program also provides for the addition of any two calculated distances. Ten of these additions are permissable.

The operator must know the sequence in which points were entered because the program identifies the points by their sequence numbers. With this identification the program can select the correct coordinate values for the distance calculations. For instance five points may be entered in the following sequence: sella, nasion, A-point, B-point, and pogonion. Therefore to find the distance between nasion and pogonion the operator must instruct the program to calculate the distance between point two and point five.

Not only do the points have sequence numbers that may go from one to one hundred, but the actual calculations have sequence numbers that go from one to a maximum of fifty. This is important for the addition of two distances option. Two calculation sequence numbers must be given for each addition of two distances. A maximum of ten such calculations are possible. For example, fifteen two point distance calculations could be made and the operator might wish to know the sum of the third distance calculation and the eleventh distance calculation. These calculations sequence numbers would be entered into the program at the appropriate time.

Following are instructions on operating and communicating with the "SIMDIS" program:

[MOUNT DSK1:]

[LOG DSK1:]

[RUN SIMDIS]

Video display sequence of program prompts:

1. The first display is a brief description and an inquiry. Respond appropriately.
2. The second is an explanation of file name entry. The data file name is the same as used in the "PLTLOD" program. Refer to the printout of that program for the exact name.
3. Next the operator will be questioned concerning the add distance option. If this option is taken, the program will need to know how many such summations will be made. Here, it is important to know beforehand what distance calculations will be made and in what sequence. The program will ask for calculation sequence number (A) and calculation sequence number (B) until it has satisfied the number of pairs to be added.
4. Whether or not the add distance option is selected, the operator will be asked how many two point distance calculations are going to be performed.
5. An instruction display follows. After it there is a display of which calculation is being performed. Then the program will ask for the point sequence numbers; two for each calculation. These are referred to as point (A) and point (B) for each calculation. Once all points are entered, the

program will perform the desired calculations on all records in the file.

6. The program will finally print out the calculated two point distances and the add distance values. Records are identified by a six digit number. There will be a final display to inform the operator that the program has run to completion.

ANGL3

This program will calculate an angle given three points. The input data file is the same as created by the "PLTLOD" program. A maximum of fifty three-point angle calculations may be made per record. As with all of the calculation programs it is important to know the sequence of the points which are on the input record file. It is with these point sequence numbers that this program references the correct coordinate values and then uses these values to determine the size of the desired angle. The calculations will be performed on all records of the input file.

Following are instructions on operating and communicating with the "ANGL3" program:

[MOUNT DSK1:]

[LOG DSK1:]

[RUN ANGL3]

Video display sequence of program prompts:

1. The first display is a brief program description.

2. Next is an explanation of the file name entry. Use the same file name as that created in the PLTLOD program.
3. Then enter the number of three point angle calculations to be made for each record.
4. The next interaction with the program is to designate the point sequence values. There are three responses per calculation. They are referred to as point (A), point (B), and point (C). When entering these values, it is important to do so in the appropriate order. Point (B) is the point at which the angle is calculated; that is the angle created between the lines defined by point(A) - point(B) and point (C) -point(B) is calculated (refer to the example in Figure 1).
5. The prescribed calculations are made on each record and the values printed out, with identification by record number. The program will display a message indicating completion of calculations.

Figure 1. To calculate the angle at nasion, angle X, the sequence values of sella, nasion, and pogonion must be entered in the following order:

point (A) - sella sequence number
point (B) - nasion sequence number
point (c) - pogonion sequence number

To calculate the angle Z, the following order is required:

point (A) - nasion sequence number
point (B) - pogonion sequence numer
point (C) - sella sequence number

To calculate the angle Y, the following order is required:

point (A) - nasion sequence number
point (B) - sella sequence number
point (C) - pogonion sequence number.

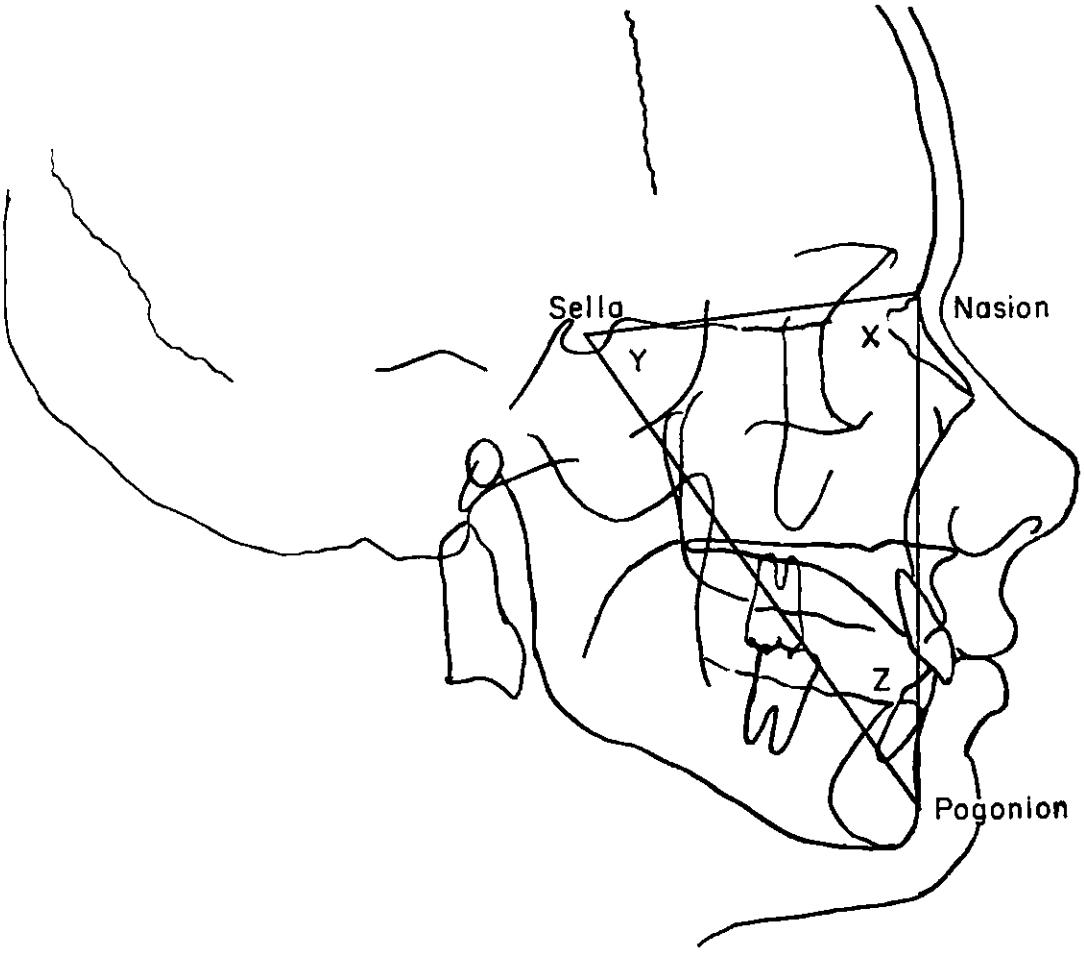


Figure 1

ANGL4

"ANGL4" is similar to the "ANGL3" program. It uses an input file, the file created by the "PLTLOD" program. Up to fifty four point angle calculations may be made in each record of the file. Again, the sequence in which the points were originally entered with the digitizer is important for point designation in this program.

Following are instructions for operating and communicating with the "ANGL4" program:

[MOUNT DSK1:]

[LOG DSK1:]

[RUN ANGL4]

Video display sequence of program prompts:

1. A short program explanation is first displayed.
2. Second, an explanation on how to enter the file name is displayed. Use exactly the same file name as that created with the "PLTLOD" program.
3. The operator should now enter the number of four point calculations that will be performed on each record.
4. Next four points will be asked for point (A), point (B), point (C), point (D). The responses to these points are the sequence values of the points that were entered in the "PLTLOD" program.

This program essentially measures an angle created by

the intersection of two lines. These two lines are defined by the four points requested by the program. Since two angles are created by the intersection of two lines, it is important that the operator enter the two points in the order which will calculate the angle desired.

One line will be defined by point (A) - point (B) and the other by point (C) - point (D). Points (A) and (C) are the points that will define which angle will be calculated. Regardless of where the two lines are on the cephalogram, the program will make the necessary computations such that the lines always intersect at point (A). The lines will still maintain their same directional relationships, however. If points (A) and (B) were reversed the complementary angle would be calculated. This is also true if points (C) and (D) were reversed (refer to the example in Figure 2).

An example of how to enter the sequence numbers in the correct points to achieve the desired calculation would be for the calculation of the Frankfort mandibular angle. Point (A) would be the sequence number for porion. Point (B) would be the sequence number for orbitale. Point (C) would be the sequence number for gonion. Point (D) would be the sequence number for menton. To change any one point would produce the calculation of the complementary obtuse angle. Remember, point (B) and point (D) are always at the ends of the two lines defining the angle to be calculated.

Figure 2. When the line formed by orbitale and porion; and the line formed by menton and gonion intersect, two angles are formed, angle X and the complementary angle Y. The sequence in which the points are entered will determine which angle is calculated.

To calculate angle X, the following order is required:

point (A) - porion sequence number
point (B) - orbitale sequence number
point (C) - gonion sequence number
point (D) - menton sequence number

To calculate angle Y, the following order is required:

point (A) - orbitale sequence number
point (B) - porion sequence number
point (C) - gonion sequence number
point (D) - menton sequence number

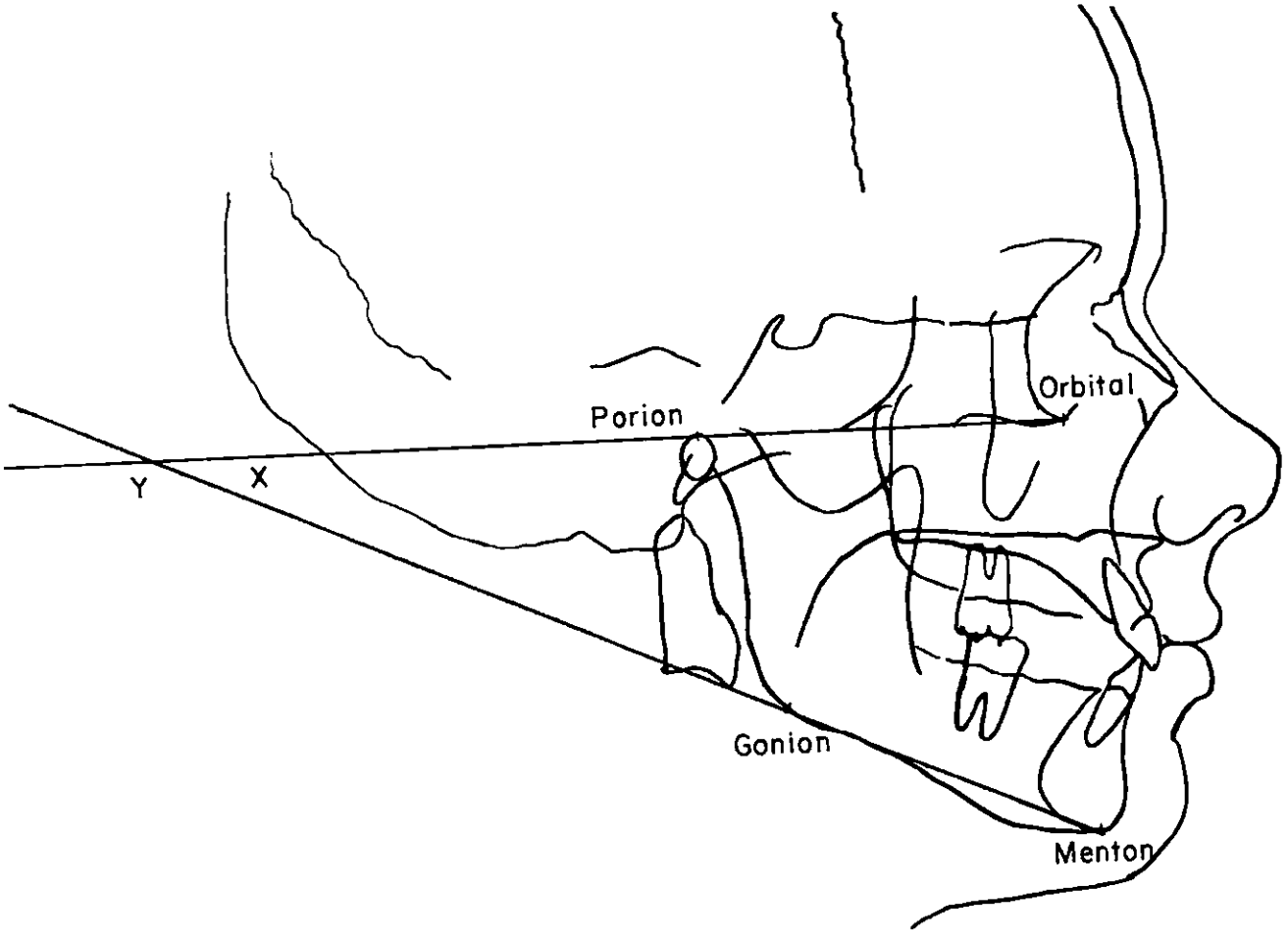


Figure 2

5. The calculations will be performed and the values printed out. The individual records are identified by a six digit number. The program will display a message indicating completion of calculations.

PTLIN

The "PTLIN" program calculates a perpendicular distance from a point to a line. It uses as an input file, the file created by the "PLTLOD" program. Fifty calculations may be performed on each record. The sequence number of the points digitized in the "PLTLOD" program are used to reference the correct coordinate values for calculations in this program. Calculations are performed on all records of the file.

Following are instructions for operating and communicating with the "PTLIN" program:

[MOUNT DSK1:]

[LOG DSK1:]

[RUN PTLIN]

Video display sequence of program prompts:

1. The first display is a brief program description.
2. Next is an explanation of the file name entry. Use the same file name as that created in the "PLTLOD" program.
3. Enter the number of calculations that will be performed on each record.

4. The program will now ask for three points: point (A), point (B), and point (C). The point sequence numbers for point (A) and point (B) will define the line. The point sequence number for point (C) will define the point from which a perpendicular distance is to be measured to the line.
5. The calculations will be performed and the values printed out, with identification by record number. A message will be displayed indicating completion of the program.

MULDST

The "MULDST" program calculates a two point distance and then multiplies that distance by a factor determined by the operator. The input file is the same as created by the "PLTLOD" program. For point designation, the sequence values of the points are used. Fifty calculations may be performed on each record. Calculations are performed on all records of the file.

Following are instructions for operating and communicating with the "MULDST" program:

[MOUNT DSK1:]

[LOG DSK1:]

[RUN MULDST]

Video display sequence of program prompts:

1. The first display is a brief program description.
2. Next is an explanation of the file name entry. Use the same file name as that created in the "PLTLOD" program.

3. Enter the number of calculations that will be performed on each record.
4. A value (X) is asked for at this time. This value may be any whole or fractional decimal value. When the value is entered, it must be entered in decimal format.
5. Two points, point (A) and point (B) are to be entered. They define a two point distance and are identified by their sequence values.
6. The calculations will be performed and the values printed out; identified by record number. A message will be displayed indicating completion of the program.

ADDIST

With this program consecutive distances may be added together. A minimum of three points which define two distances and a maximum of five points that define four distances can be used. The input file is the same as created by the "PLTLOD" program. The points are identified by their original sequence numbers. Up to fifty calculations can be performed on each record. All records of the file will be acted upon for computations.

Following are instructions for operating and communicating with the "ADDIST" program:

[MOUNT DSK1:]

[LOG DSK1:]

[RUN ADDIST]

Video display sequence of program prompts:

1. The first display is a brief program description.
2. The second display is an explanation of file name entry. Use the same name as that created in the "PLTLOD" program.
3. Enter the number of calculations to be performed on each record.
4. There is now an inquiry as to the number of points for calculation. This question is asked for each calculation. This allows for variability of the number of points used in distance calculations within the record. For example if six calculations were to be made on each record, the first two might involve four points each, the next three, five points, and the last calculation, three points.
5. When the number of points for calculations has been entered, the program will react accordingly and ask for points (A), (B), and (C). Depending on the value of points for calculation points (D) and (E) may be asked for. The response to all point inquiries will be the sequence values of the points to be used in the distance calculations.
6. Calculations will be performed and values printed out with identification by record number. A message will be displayed indicating completion of the program.

Program Information

All of the previously discussed programs have been

designed with similarity in mind. If for any reason the operator wishes to terminate a program, the value zero should be entered when point (A) is requested. The same result may be accomplished at any place within the operation of the program by holding down the CONTROL key and at the same time pressing the C character key.

If any problem occurs when the points are being entered, there is a displayed inquiry to ask if all point values are correct. By answering no, [N], to this question, the program will not complete the acceptance of the present series of points but will go back and start at point (A) again. By answering yes, [Y], the program will continue and ask for the next series of points.

CANLOD

"CANLOD" is a program that is similar to "PLTLOD". The main difference is that this program was designed more with clinical application in mind than research application. There is considerably less variability allowed in this program and the analysis program which uses the output file.

For each record, a six digit identification number, a last name and a first name are required. A total of sixteen points will be entered for each cephalogram. The points to be entered are as follows and must be entered in the order specified by the number preceeding them:

1. Sella
2. Nasion

3. Orbitale
4. A-point
5. B-point
6. Pogonion
7. Gnathion
8. Menton
9. Apical of upper 1 - the apex point of the representative maxillary central incisor
10. Incisal of upper 1 - the incisal point of the representative maxillary central incisor
11. Incisal of lower 1 - the incisal point of the representative mandibular central incisor
12. Apical of lower 1 - the apex point of the representative mandibular central incisor
13. Incisor difference point - that point which divides the amount of incisor overbite or open bite by one half
14. Molar point - mesial buccal cusp of maxillary first molar
15. Gonion
16. Porion

Approximately three hundred records may be loaded onto a disc that has no other data files already on it.

Following are instructions on operating and communicating with the "CANLOD" program:

[MOUNT DSK1:]

[LOG DSK1:]

[RUN CANLOD]

Video display sequence of program prompts:

1. The first display is a brief program description.
2. Next there will be a print out indicating which points must be entered and in what order they should be entered. Simultaneously, instructions concerning the output file name are displayed on the video screen.
3. This is a request for the disc number that may be found on the disc label.
4. A six digit identification, last name, and first name will be entered at this time.
5. Instructions concerning error are displayed and the operator is instructed to enter the points from the digitizer. To erase a point, press the stylus in the right hand boundary area. To exit from the program, reset the program, or restart point entry for a cephalogram press the stylus in the upper boundary area.

   After the stylus has been depressed in the right hand boundary area, the next entry point will be displayed. If the stylus is pressed in the upper boundary area, the first question asked is if the operator wishes to exit from the program. If the answer is no, [N], the program will ask about resetting the program. If this option is accepted the program is essentially begun again and all records previously recorded are lost. If the reset option is not selected, the

operator has the option, then, of restarting the digitizing of the cephalograms.

6. After each cephalogram has been digitized the program will ask the operator if there is another one to be entered. Answer [Y] for yes or [N] for no.
7. The program, upon completion, will print out the file name, the disc number, and the number of cephalograms digitized.

CANNAL

"CANNAL" is the most complex program in the present system. However, from the operator's point of view, it is the easiest to use. As its name implies, the program performs a "canned" analysis on all of the records of the file created by the "CANLOD" program. The program will perform any combination of the following analyses: Tweed, Downs, Begg and Reidel.

Following are instructions on operating and communicating with the "CANNAL" program:

[MOUNT DSK1:]

[LOG DSK1:]

[RUN CANNAL]

Video display sequence of program prompts:

1. A series of four questions occur at this point in the program. The program will ask if the operator wants the Tweed analysis performed, the Downs analysis performed, the Begg analysis performed, or the Reidel analysis

performed. The operator has only to answer yes, [Y], to the ones desired and no, [N], to those not required.

2. Next is a request to enter the input file name. Enter exactly what was used for the file name when the file was built with the "CANLOD" program.
3. The program will print out the desired results and indicate when processing has been completed.

DISCUSSION

The objectives of this project, as previously mentioned were to design a microcomputer configuration; select hardware; select a distributor; determine applications; provide programs; write an operator's manual; and make recommendations for others interested in microcomputer systems.

In the present case, the present investigator has had programming and systems design experience with the RCA Spectra 70/45 and 70/60 computer systems. Programming experience was with assembler language and COBOL. There was no previous experience with microcomputer systems or with the BASIC language.

The ensuing discussion will be divided into two sections. The first will contain a proposed approach for the selection through implementation process. The recommendations are based upon information obtained during the project. The second will evaluate the project as a whole and problems encountered.

Proposed Approach

Currently, articles dealing with microcomputer systems and their potential applications in dental offices have appeared frequently in the dental literature. Dental product displays at conventions usually have several companies representing the microcomputer industry. Dentists are becoming progressively more

interested in microcomputers, and it is important that they critically evaluate the legitimate literature and marketing literature of microcomputer companies.

Before the dentist considers any possibility of installing a computer system in his office, he should have a very thorough understanding of his certain relevant operations and needs involved in practice. This includes not only what is done by the dentist but also the personnel who are in his employ. The dentist should know what business functions are done in the office and what the procedures are for doing them.

If these criteria have been fulfilled the dentist is in a better position to judge how well his practice is organized. If relatively poorly organized, there is no need to consider a microcomputer. A computer can only be effective in a highly organized environment.

At least an elementary knowledge of what a computer is, how it functions, and precisely what it can do is necessary. There are a number of books available that can introduce the reader to computer terminology and concepts. Two such books are Minicomputers by Donald P. Kennedy and Basic Basic by James J. Coan. With this information and an understanding of office records and business functions, the dentist can start to evaluate the practice in terms of computer applications.

Not just any computer system will work in a practice. Nor will any one computer system fulfill all contingencies in all

practices. There are some practice-related considerations to be cognizant of when evaluating different systems. Size of practice, in terms of patients, doctors, and adjunct personnel, is important. Type of practice, i.e., group, multispecialty, solo, incorporated, partnership, is an important consideration. The dentist should have an idea of which business functions would be appropriate for computer implementation. These functions should also have priorities as to desirability of implementation. If records related to patient treatment and medical-dental history are to be kept on the computer, the record sizes and formats should be considered.

Once the above preliminary work has been accomplished, it is time to evaluate different computer systems. The first consideration is the hardware. The hardware should be analyzed in terms of capability relative to previously established objectives. The hardware should provide the necessary functions and still be versatile enough that it could be upgraded for future system growth. It is best to start slowly but leave open avenues for future development. Extreme austerity or extravagance will likely result in a poor cost-effectiveness ratio.

The software is equally as important a consideration as the hardware. It should be able to support all of the hardware necessary to perform the desired jobs. Operating system software should provide for the desired language options. Utility software can be very helpful in maintaining a system and should be available. If prewritten application software exists it should be checked for its ability to fulfill processing needs.

Documentation, written information specific for hardware and software is often the weak link in computer systems. Without good documentation it will be impossible to ever achieve maximum system potential. Documentation should be available for study prior to any purchase agreement.

Costs are always a relevant factor. Hardware costs are related to the sophistication of the hardware and to the number of pieces of equipment. Some software costs are included in the price of the equipment; however, developmental costs must be applied to individualized program systems. Training is a cost to be considered. Conversion costs will be incurred when changing to an automated information processing system. Maintenance and upkeep are factors that increase the overall cost of a system. Operating costs must certainly be added.

There is a time factor involved in selecting a computer system. From the time of inception of the idea of microcomputer processing to the first production runs may require several months to several years. The time requirement would of course be less if the aid of professional systems personnel were employed.

Time must also be allocated on a daily basis for systems operation. This involves an initial learning time for the operator. After this phase is accomplished, production time is then the primary factor.

Back-up systems become a necessity at the time that production begins. When a potential user is evaluating systems the

subject of back-up procedures has to be carefully considered. File back-up procedures must be incorporated into program operations. Equipment back-up should be arranged with the supplier. This is especially important once an office becomes dependent upon a computer system. Alternative methods should be in back-up procedures in the event of a serious problem with the microcomputer that shuts down production for any period of time.

Selecting and implementing a computer system is obviously not a simple task. It requires much thought, study, time and hard work to become efficiently operational. It would be difficult for a dentist untrained in computer science to develop this proficiency within an economical or realistic time frame. Therefore, it is advisable to secure the assistance of an independent systems consultant. This would of course be an additional cost, but in the long run the savings could be significant.

In a well organized office a functional microcomputer system can be a very worthwhile adjunct that provides many valuable assets. There is a decrease in the amount of space needed to store records or files. Current business account information may be available for immediate inspection. Retrieval time for scheduling information patient file processing, and business account processing can be diminished. Insurance form processing is more easily done by computer than manually. Word processing is a very efficient method for written communication. Business analyses can be more completely done by computer than by individuals and at

considerably lower cost. In large practices, it may be possible to reduce the number of man-hours necessary to operate the office. The microcomputer can also reduce the amount of mundane and repetitive type of procedures, thus making an office a more enjoyable place in which to work.

Project

This project originally started as an investigation into the feasibility of obtaining and utilizing some data processing equipment for the Orthodontic Department. Some equipment of a stand-alone form was initially evaluated, and it was determined that the department would be better served with a microcomputer system.

A few suppliers were visited and the capabilities of some of their systems were discussed. Standard advertising brochures and portfolios were obtained and studied. The availability of installation and set-up of assistance as well as maintenance was discussed with the suppliers.

During this period, there were numerous ideas and proposals related to the types and operation of microcomputer systems. As plans grew, so did a realization for the need of various equipment pieces to facillitate future planned projects. Additional equipment in terms of peripherals was proposed and some was purchased.

The equipment was installed and a familiarization stage was begun. This involved learning how to operate the equipment and

how to use the operating system and programming manuals. To assist in this phase, an employee of the supplier was hired to provide some initial training.

The decision was made at a later date to concentrate on the development of a digitizing system. This system was to enable an operator to digitize cephalometric points. From such coordinate values, programs were developed to calculate two point distances, three point angles, four point angles, multiple distances, distances multiplied by factors, and perpendicular distances from a point given to a line.

The programs to provide the above mentioned capabilities were designed, written and tested. These particular programs were designed with research application in mind.

A need was seen for an analysis program that could be used as a clinical tool in diagnosis and treatment planning. Two appropriate programs were designed, one to digitize cephalograms, and one to perform the analyses. The analysis program enables a user to perform any one, or combination, of all of the following analyses: Tweed, Downs, Begg, and Reidel.

The selection of processing hardware and software was based on the precepts that the processor must have very fast instruction execution time; have multi-user capability; and that the software be of high sophistication.

The graphics display terminal was chosen because it was thought that it would be desirable to be able to display a

cephalogram from disc storage onto a high resolution video terminal. The possibility of growth prediction and growth projection was considered. The graphics terminal was at first also to be used as a control console.

The printer that was selected was on the recommendation of the supplier. It was advised by the supplier that this particular printer was adequate to provide the necessary capabilities.

A digitizing board was also placed on early order. It was recommended by the supplier, although there was only a small amount of printed information at the time describing the board.

The comments below are a result of a retrospective evaluation of our own steps of progression looking at different computer systems and then the purchase of a system. These comments are offered to provide the reader with insight as to pitfalls when considering such a purchase.

It is the opinion here that there was too much impulse from the inception of the idea to implement a computer system to the purchase of the hardware. Although a greater amount of time and effort would have been required, the first step should more properly have been a more thorough analysis of departmental functions and a determination of specific applications for microcomputer implementation. An application priority list might also have been compiled at this time.

The next level of involvement should have involved more thorough investigation and research into general microcomputer

capabilities. This would have established a stronger basis for evaluation of the different available systems.

At this point, two activities should have occurred. An independent systems analyst with microcomputer experience should have been consulted with reference to a hardware configuration and manufacturer recommendation. Supplier recommendation might have been obtained also. Detailed and complete documentation of the potential hardware and software should have been obtained for thorough evaluation.

By consulting a systems analyst, a more realistic approach and timetable could have been established. The pros and cons of self development versus outside development could also have been better evaluated.

Difficulties were encountered almost immediately after taking delivery of the system. Approximately one month of orientation time was lost because an incomplete operating system disc was supplied by a local distributor.

In addition to an incomplete operating system, two other early problems were encountered. Both have proven to be factors that detrimentally affected the development of user systems. One is the complete inadequacy of documentation. When a user is developing a system, documentation as to how to use the operating software is of utmost importance. Learning the systems capabilities was reduced to uninformed trial and error tactics.

The second major problem has been that of malfunctioning

equipment. Problems, a number of them serious and which delayed progress for several months, were experienced with the central processing unit and the disc drive. Later, after the acquisition of a different console terminal, physical difficulties also occurred with this replacement unit.

The operating systems, with all of its professed capabilities, is somewhat bulky. It was mentioned that some of the advertised capabilities exist on paper but not in reality, or are at least shrouded beyond recognition. Two software inadequacies have made using the system difficult. One is the almost total lack of error recovery. The other is the inability to use memory overlays.

With most computer systems, revisions which update and enhance the system are published on a regular basis. There have been none available since our particular system was obtained. This problem could lie with either the system manufacturer or the supplier, or both.

The development of application programs and systems is a demanding process. However, once the programs have been written and tested, the dentist can process large quantities of information with relative ease. Information processing can be more accurate and can be done at a fraction of the time required within a manual system.

The programs designed for the Case Western Reserve University Orthodontic Department are very useful for research or clinical application. Very little time is required to learn how to use the computer equipment or the programs. The ease of use of the

system and the speed at which information may be accurately processed has been especially helpful to graduate students doing research involving the use of cephalograms. In most cases, several weeks worth of work were reduced to four or five hours of work.

With the low cost of microcomputer systems, the potential for use in dental offices and dental school departments certainly exist. In a busy practice or department microcomputers can open new avenues of information processing for business, diagnosis and treatment planning, and research.

Complexities and problems can be great, but are not insurmountable Good organization and planning are absolutely necessary to reduce the complexities and circumvent possible major problems. Based upon the experiences obtained from the present project, it is the judgement of this investigator that the process of selection, design, and implementation of a microcomputer system is beyond the scope of most dental practitioners, given reasonable but not unlimited effort. The time factor alone makes it unrealistic for all but the hobbyist. This does not mean that the dentist should not be involved. Quite the contrary, he should make every effort to become knowledgeable, but should not expect to become a systems analyst.

SUMMARY AND CONCLUSIONS

Computer systems in the form of microcomputers have made electronic information processing an economic feasibility for small organizations. Two organizations which can derive benefits from the microcomputer are a dental office and a department within a dental school.

Microcomputers may be used for business functions. They cal also be used in clinical application or for research purposes. Unfortunately, there are few dentists who have the background to evaluate their needs for a computer system and to then proceed with with procurement and implementation of an appropriate system.

The purposes of this study are as follows:

1. Design a microcomputer hardware configuration for the Orthodontic Department of the Case Western Reserve University Dental School.
2. Select the specific types of hardware and their manufacturers.
3. Select a qualified and competent distributor.
4. Determine applications and their priority for the Department of Orthodontics at the Case Western Reserve University Dental School.
5. Design, write, and implement programs for the selected microcomputer system that could be used for clinical or research purposes.
6. Provide an operations manual that would enable a student, dentist, or auxillary to use the system.

7. Evaluate the entire process and provide pertinent comments and recommendations for others who might be interested in microcomputer systems.

A six component system was configured for the Orthodontic Department. Using this system, nine programs were developed. Seven of those were created for research purposes and two primarily for clinical application. An operations manual was written to facilitate the use of the programs.

In addition, suggested recommendations for dentists interested in microcomputer systems have been set forth.

LITERATURE CITED

Alpha Microsystems. 1977. AlphaBasic-User's Manual. Irvin, California: Alpha Microsystems, 69 pp.

Alpha Microsystems. 1979. AMOS Users Guide. Irvine, California: Alpha Microsystems, 132 pp.

Alpha Microsystems. 1979. Release Notes Version 4.Ø. Irvine, California: Alpha Microsystems, 97 pp.

Alpha Microsystems. 1979. Vue: Screen-Oriented Text Editor-Version 2.0. Irvine, California: Alpha Mircosystems, 35 pp.

Bebey, G. A., and Schwartz, M.D. 1972. Hospital Information Systems. New York: Marcel Dekker, Inc., 402 pp.

Bergin, R., Hallenberg, J., and Malmgren, O. 1978. Computerized Cephalometrics. Acta Odontol. Scand., 36(6): 349-357.

Brigham, C. 1973. Minicomputers in Health Science Instruction. Springfield, Virginia: National Technical Information Service, 34 pp.

Clifton, H. D. 1975. Choosing and Using Computers. London: Business Books, Ltd., 205 pp.

Coan, J. S. 1978. Basic Basic. Rochelle Park, N. J.: Hayden Book Company, Inc., 269 pp.

Coburn, H. E. 1979. Concerning User's Manuals. Byte, June: 190-192.

Colsher, W. L. 1978. Backup techniques. Kiloband, 15: 74-75.

Corby, C. S. 1978. Are you ready for a computer? Dental Econ., August: 32-36.

Economides, J. K. 1979. A microcomputer in the orthodontic office. J. C. O., 13(2): 767-772.

Erlich, A. 1979. Is there a computer in your future? Dental Graduate, June: 57-60.

Goldman, A. R. 1978. Make way for the in-office computer. Dental Management, March: 31-47.

Halsema, A. I. 1979. Bubble memories: A short tutorial. Byte, June: 166-167.

Kenney, D. P. 1978. Minicomputers. New York: AMACOM, 269 pp.

Liben, S. 1978. The first ten years of amateur computing. Byte, July: 64-71.

Orear, J. 1967. Fundamental Physics. New York: John Wiley and Sons, Inc., pp. 369-372.

RCA Corporation. 1967. Introduction to Electronic Data Processing. Document # 70-00-801. Camden, N. J.: Customer Education Department, Information Systems Division, RCA Corporation, 266 pp.

RCA Corporation. 1969. Introduction to Spectra 70 Programming. Document #70-00-803. Camden, N. J.: Marketing Education Department, Information Systems Division, RCA Corporation, 138 pp.

Reid-Green, K. S. History of Computers: The IBM 704. Byte, January: 190-192.

Reid-Green, K. S. 1978. A short history of computing. Byte, July: 84-94.

Ricks, M. 1980. A computer helped humanize my office. Dental Management, 20(3): 18-24.

Roedema, W. H., Knapp, J. G., Spencer, J., and Dever, M. K. 1978. Computer-graphic representation of mandibular movements in three dimensions. Part II: The sagittal and frontal planes. J. Prosthetic Dent., 40(4): 385-391.

Sparks, F. W., and Rees, P. K. 1965. Plane Trigonometry with Tables. Englewood Cliffs, N. J.: Prentice-Hall, Inc., 306 pp.

Siler, W. and Lindberg, D. A. B. 1974. Computers in Life Science Research. New York: Plenum Publishing Corp., 272 pp.

APPENDIX I

PLTLOD.BAS - Program and Flowchart

File DSK1:FLTLOD.BAS[1,4] Printed on 6/6/80

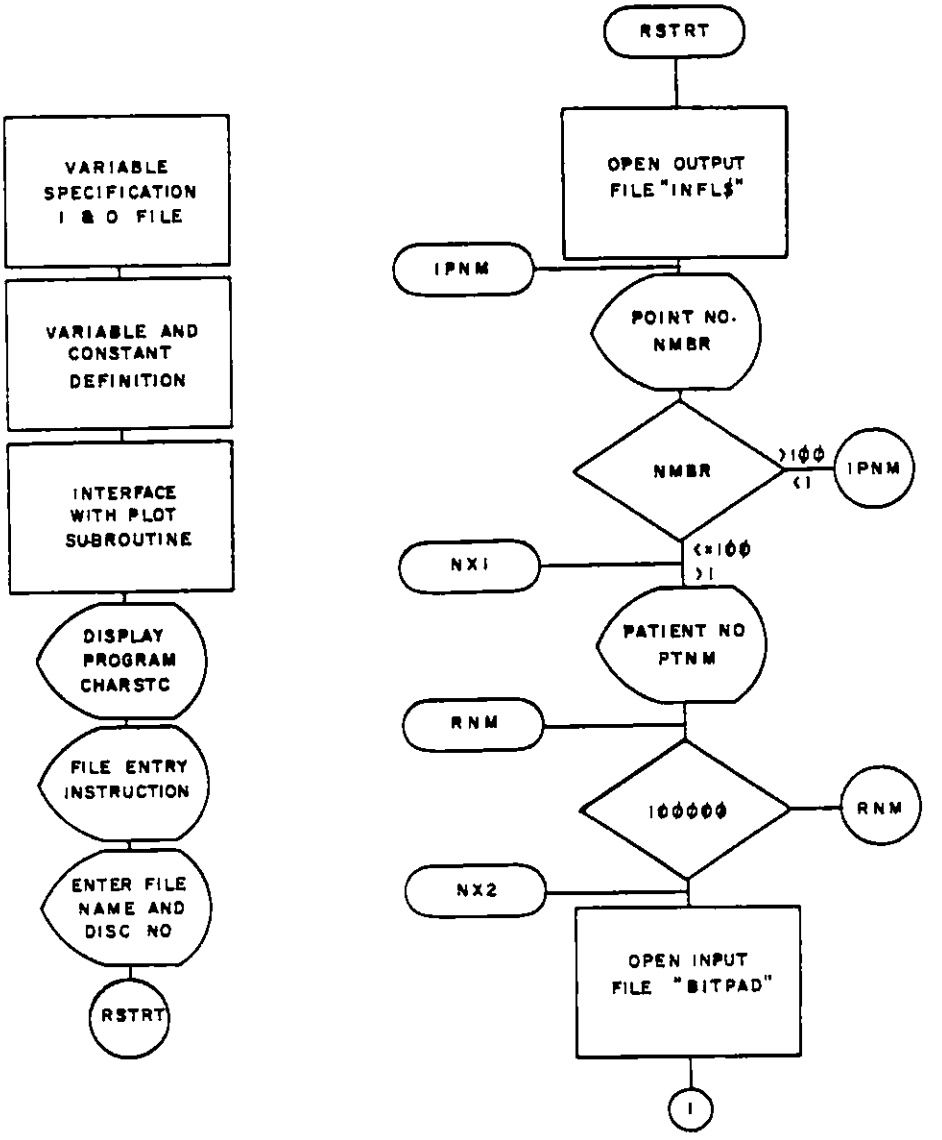
```
!          ********************************************************
!          *                                                      *
!          *    FLTLOD.BAS - THIS PROGRAM WILL TRANSFER           *
!          *    CEPHALOMETRIC COORDINATES FROM THE DIGITIZER      *
!          *    TO A DISC FILE.                                   *
!          *                                                      *
!          ********************************************************
!
!
!
          STRSIZ 20
          MAP1 XY
             MAP2 X,B,2
             MAP2 Y,B,2
             MAP2 FLAG,B,2
          MAP1 XYOUT(100)
             MAP2 XOUT,B,2
             MAP2 YOUT,B,2
          MAP1 PTNMB
             MAP2 PTNM,F,6
          MAP1 CEPT
             MAP2 CEPT,F,6
!
!
          CEPNM = 0
          SUB1 = 1
          LET FL$ = "00000"
!
          XCALL PLOT,"TRMB"
!
!
          PRINT TAB(-1,0)
          PRINT "THIS PROGRAM WILL TRANSFER CEPHALOMETRIC COORDINATES FROM"
          PRINT "THE DIGITIZER TO A DISC FILE. IT WILL BE THE RESPONSIBILITY"
          PRINT "OF THE OPERATOR TO KNOW IN WHAT SEQUENCE THE POINTS"
          PRINT "WERE ENTERED. A MAXIMUM OF 100 POINTS PER CEPH MAY"
          PRINT "BE ENTERED. A MINIMUM OF 2 POINTS PER CEPH MUST BE ENTERED."
          PRINT "A 6-DIGIT IDENTIFICATION NUMBER IS REQUIRED FOR EACH"
          PRINT "CEPH. THIS WILL BE ENTERED FROM THE CONSOLE PRIOR TO"
          PRINT "DIGITIZING A CEPH. A MAXIMUM OF 90 CEPHS MAY BE"
          PRINT "TRANSFERED TO ANY ONE DISC."
          PRINT " "
          INPUT "PRESS RETURN TO CONTINUE:  ",I$
          PRINT TAB(-1,0);TAB(5,5)
          PRINT "ENTER THE NAME OF THE DATA FILE THAT IS TO BE";TAB(6,5)
          PRINT "CREATED. THE NAME CAN BE ANY ALPHA-NUMERIC";TAB(7,5)
          PRINT "COMBINATION, BUT MUST START WITH A LETTER.";TAB(8,5)
          PRINT "THE FILE NAME MUST BE PRECEDED BY DSK1:.";TAB(9,5)
          PRINT "THE NAME MUST BE IMMEDIATELY FOLLOWED WITH";TAB(10,5)
          PRINT "A PERIOD AND THE THREE LETTERS DAT. FOR";TAB(11,5)
          PRINT "EXAMPLE TO NAME A FILE CEPH ONE SHOULD";TAB(12,5)
          PRINT "ENTER DSK1:CEPH.DAT FOR THE FILE NAME.";TAB(13,5)
          INPUT "ENTER THE FILE NAME : ",INFL$
          PRINT " ";TAB(15,5)
          INPUT "ENTER THE DISC NUMBER : ",DISCNM
!
RSTRT :   OPEN #5,INFL$,OUTPUT
IPNM :    PRINT TAB(-1,0);TAB(10,5)
          INPUT "ENTER THE NUMBER OF POINTS TO BE TRANSFERED: ",NMBR
          LET CEPT = NMBR
```

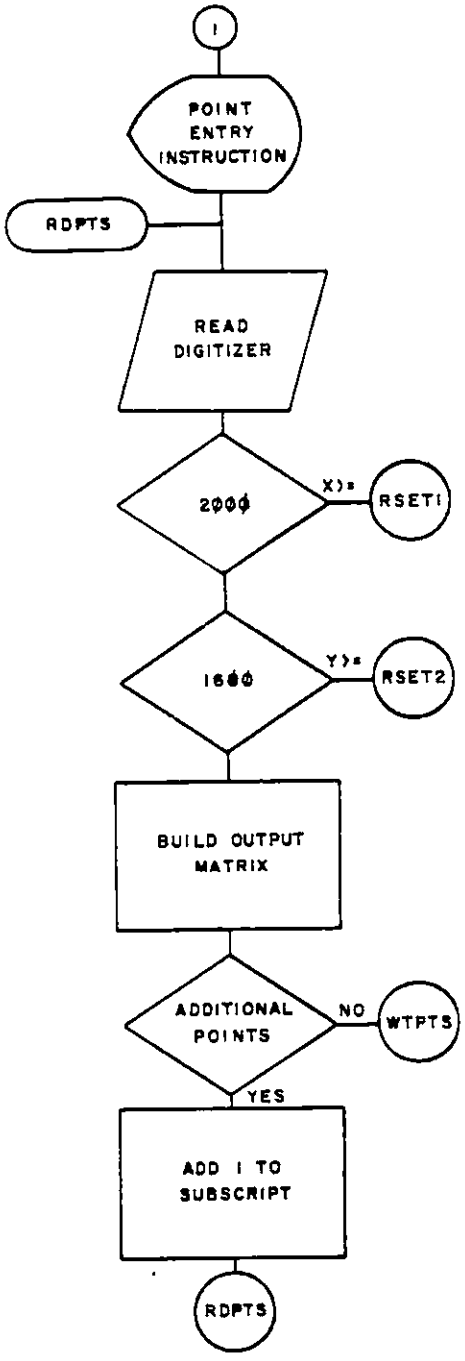
```
File DSK 1:FLTLOD.BAS[1,4] printed on 0/0/0

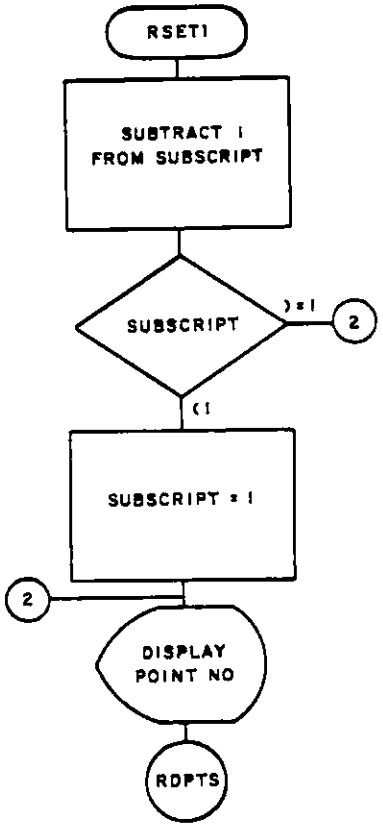
          IF NMBR >= 100 AND NMBR <1 GOTO NX1
          GOTO IFNM
!
!
NX1':     PRINT TAB(-1,0);TAB(5,5)
          PRINT "MOUNT CEPH TRACING, LINE UP THE EDGES WITH ";TAB (6,5)
          PRINT "THE MARK ON THE DIGITIZER. ";TAB(7,5)
          PRINT "A 6-DIGIT PATIENT NUMBER MUST NOW BE ";TAB(8,5)
          PRINT "ENTERED. THE LEFT MOST DIGIT IS THE SEQUENCE";TAB(9,5)
          PRINT "NUMBER WHEN A PATIENT MAY HAVE MORE ";TAB(10,5)
          PRINT "THAN ONE CEPH. THIS NUMBER CANNOT BE ";TAB(11,5)
          PRINT "LESS THAN (1).";TAB(15,5)
RNM :     INPUT "ENTER 6-DIGIT PATIENT NUMBER :",PTNM
          IF PTNM <= 100000  GOTO NX2
          GOTO RNM
!
!
NX2 :     OPEN #3,"TRM:BITREAD",INPUT
          PRINT TAB(-1,0);TAB(5,5)
          PRINT "POINTS MAY ONLY BE RECORDED FROM WITHIN THE ";TAB(6,5)
          PRINT "BOUNDARIES OF THE MARKS ON THE DIGITIZER. IF";TAB(7,5)
          PRINT "A MISTAKE IS MADE, PRESS THE STYLUS TO THE";TAB(8,5)
          PRINT "RIGHT OF THE RIGHT-HAND BOUNDARY.";TAB(9,5)
          PRINT "THIS WILL CAUSE THE PREVIOUS ENTRY TO BE";TAB(10,5)
          PRINT "ERASED. TO RESET THE PROGRAM, RESTART A";TAB(11,5)
          PRINT "TRACING, OR EXIT FROM THE PROGRAM PRESS";TAB(12,5)
          PRINT "THE STYLUS ABOVE THE UPPER BOUNDARY. POINTS";TAB(13,5)
          PRINT "MAY NOW BE ENTERED."
!
RDPTS :   INPUT #3,X,Y,FLAG
          IF X >= 2000 GOTO RSET1
          IF Y >= 1600 GOTO RSET2
          LET XOUT(SUB1) = X
          LET YOUT(SUB1) = Y
          IF SUB1 = NMBR GOTO WTPTS
          SUB1 = SUB1 + 1
          GOTO RDPTS
!
RSET1 :   LET SUB1 = SUB1 - 1
          IF SUB1 <= 1 LET SUB1 = 1
          PRINT TAB(15,5);TAB(-1,9);TAB(16,5);
          PRINT "YOU ARE NOW ENTERING POINT # :";SUB1
          GOTO RDPTS
!
RSET2 :   LET SUB1 = 1
          PRINT TAB(-1,0);TAB(10,5)
          INPUT "DO YOU WISH TO EXIT FROM THE PROGRAM (Y OR N) :",AN1$
          IF AN1$ = "Y" GOTO ENDRT
          PRINT TAB(-1,0);TAB(10,5)
          INPUT "DO YOU WISH TO RESET THE PROGRAM (Y OR N) :",AN1$
          IF AN1$ = "N" GOTO RECEF
          CLOSE #3
          CLOSE #5
          CEFNM = 0 : NMBR = 0 : PTNM = 0
          GOTO RSTRT
RECEF :   PRINT TAB(-1,0);TAB(10,5)
          INPUT "DO YOU WISH TO RESTART THE TRACING (Y OR N) :",AN1$
          IF AN1$ = "N" GOTO RSET2
          CLOSE #3
          GOTO NX1
```

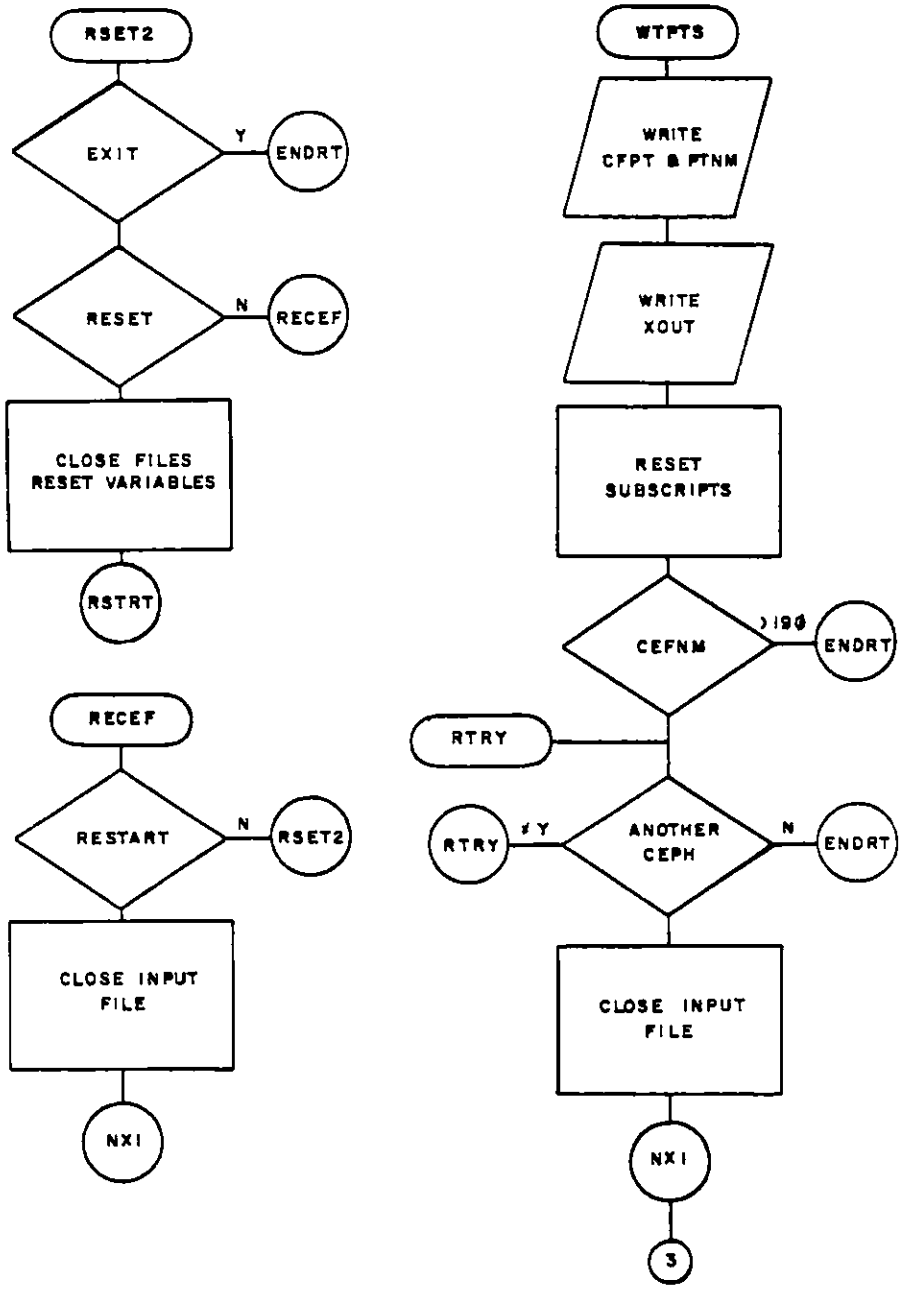
File DSK1:FLTLOD.BAS[1,4] printed on 0/0/0

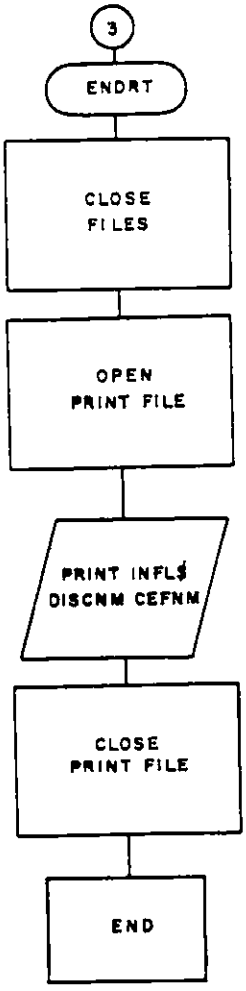
```
!
!
WTPTS :    PRINT #5,CFPT,FTNM
           SUB1 = 1
           FOR I =1 TO 100
           PRINT #5,XOUT(SUB1);YOUT(SUB1)
           LET SUB1 = SUB1 + 1
           NEXT I
           FOR I = 1 TO 20
           PRINT #5,FL$
           NEXT I
           LET CFFNM = CFFNM + 1
           LET SUB1 = 1
           IF CFFNM <> 90 GOTO ENDRT
RTRY :     PRINT TAB(-1,0);TAB(5,5)
           INPUT "DO YOU WISH TO ENTER ANOTHER CFFH (Y OR N) :",ANS$
           IF ANS$ = "N" GOTO ENDRT
           IF ANS$ <>  "Y" GOTO RTRY
           CLOSE #3
           GOTO NX1
!
ENDRT :    CLOSE #3
           CLOSE #5
           OPEN #7,"TRM:PRINT",OUTPUT
           PRINT #7,CHR(12)
           PRINT #7,TAB(40);"THE NAME OF YOUR FILE IS :";INFL$
           PRINT #7,TAB(40);"YOUR DISC NUMBER IS";DISCNM
           PRINT #7,TAB(40);"YOU HAVE TRANSFERED";CFFNM;"CFFHS TO THE DISC"
           CLOSE #7
           END
```


VARIABLE SPECIFICATION I & O FILE
VARIABLE AND CONSTANT DEFINITION
INTERFACE WITH PLOT SUBROUTINE
DISPLAY PROGRAM CHARSTC
FILE ENTRY INSTRUCTION
ENTER FILE NAME AND DISC NO
RSTRT
RSTRT
OPEN OUTPUT FILE "INFL$"
IPNM
POINT NO. NMBR
NMBR
>100
<1
IPNM
NX1
<=100
>1
PATIENT NO PTNM
RNM
100000
RNM
NX2
OPEN INPUT FILE "BITPAD"
1


1
POINT ENTRY INSTRUCTION
RDPTS
READ DIGITIZER
2000
X>=
RSET1
1600
Y>=
RSET2
BUILD OUTPUT MATRIX
ADDITIONAL POINTS
NO
WTPTS
YES
ADD 1 TO SUBSCRIPT
RDPTS


RSET1
SUBTRACT 1 FROM SUBSCRIPT
SUBSCRIPT
>=1
2
<1
SUBSCRIPT = 1
2
DISPLAY POINT NO
RDPTS


RSET2
EXIT
Y
ENDRT
RESET
N
RECEF
CLOSE FILES
RESET VARIABLES
RSTRT
RECEF
RESTART
N
RSET2
CLOSE INPUT
FILE
NXI
WTPTS
WRITE
CFPT & FTNM
WRITE
XOUT
RESET
SUBSCRIPTS
CEFNM
)190
ENDRT
RTRY
RTRY
≠Y
ANOTHER
CEPH
N
ENDRT
CLOSE INPUT
FILE
NXI
3


3
ENDRT
CLOSE
FILES
OPEN
PRINT FILE
PRINT INFL$
DISCNM CEFNM
CLOSE
PRINT FILE
END

APPENDIX II

SIMDIS.BAS - Program and Flowchart

File DSK1:SIMDIS.BAS[1,4] printed on 0/0/0

```
!          ************************************************
!          *                                              *
!          *   SIMDIS.BAS - THIS PROGRAM WILL CALCULATE   *
!          *   A DISTANCE BETWEEN TWO POINTS AND ADD TWO  *
!          *   DISTANCES TOGETHER.                        *
!          *                                              *
!          ************************************************
!
          STRSIZ 20
          MAP1 FNBR
             MAP2 FNB,F,6
          MAP1 XYSTR(100)
             MAP2 XSTR,B,2
             MAP2 YSTR,B,2
          MAP1 AB(50)
             MAP2 A,F
             MAP2 B,F
          MAP1 LIN(50)
             MAP2 IN,F,6
          MAP1 CEPT
             MAP2 CEPT,F,6
          MAP1 FILR
             MAP2 FILL,B,5
          MAP1 CALCA(10)
             MAP2 CALC2,F
             MAP2 CALC3,F
          MAP1 ADDIST(10)
             MAP2 ADD1,F
          FLG = 0
!
!
START :   PRINT TAB(-1,0)
          PRINT "THIS PROGRAM IS FOR THE CALCULATION OF A DISTANCE BETWEEN"
          PRINT "TWO POINTS. THE DESIRED CALCULATIONS WILL BE PERFORMED"
          PRINT "ON ALL RECORDS OF THE FILE. A MAXIMUM OF (50) CALCULATIONS"
          PRINT "MAY BE EXECUTED PER RECORD. THIS PROGRAM ALSO PERMITS THE"
          PRINT "ADDITION OF ANY TWO DISTANCES. TEN OF THESE CALCULATIONS"
          PRINT "MAY BE MADE PER RECORD."
          PRINT " "
          INPUT "DO YOU WISH TO USE THIS OPTION (Y OR N) :",AN1$
          IF AN1$ = "N" GOTO EN
          IF AN1$ = "Y" GOTO CONT1
          GOTO START
CONT1 :   PRINT TAB(-1,0);TAB(5,5)
          PRINT "ENTER THE NAME OF THE DATA FILE THAT IS TO BE USED.";TAB(6,5)
          PRINT "THE FILE NAME MUST BE PRECEDED BY DSK1:.";TAB(7,5)
          PRINT "THE NAME MUST BE IMMEDIATELY FOLLOWED WITH";TAB(8,5)
          PRINT "A PERIOD AND THE THREE LETTERS DAT. FOR EXAMPLE";TAB(9,5)
          PRINT "IF THE FILE NAME IS CEPH ONE SHOULD ENTER";TAB(10,5)
          PRINT "DSK1:CEPH.DAT FOR THE FILE NAME. YOUR FILE";TAB(11,5)
          PRINT "NAME WILL BE THE SAME AS CREATED BY THE";TAB(12,5)
          PRINT "FLTLOD PROGRAM.";TAB(13,5)
          INPUT "ENTER THE FILE NAME : ",INFL$
!
          OPEN #5,INFL$,INPUT
          OPEN #7,"TRM:PRINT",OUTPUT
          PRINT #7,CHR(12)
          PRINT #7,TAB(40);"TWO POINT DISTANCE CALCULATIONS"
DISOF :   PRINT TAB(-1,0);TAB(15,5)
```

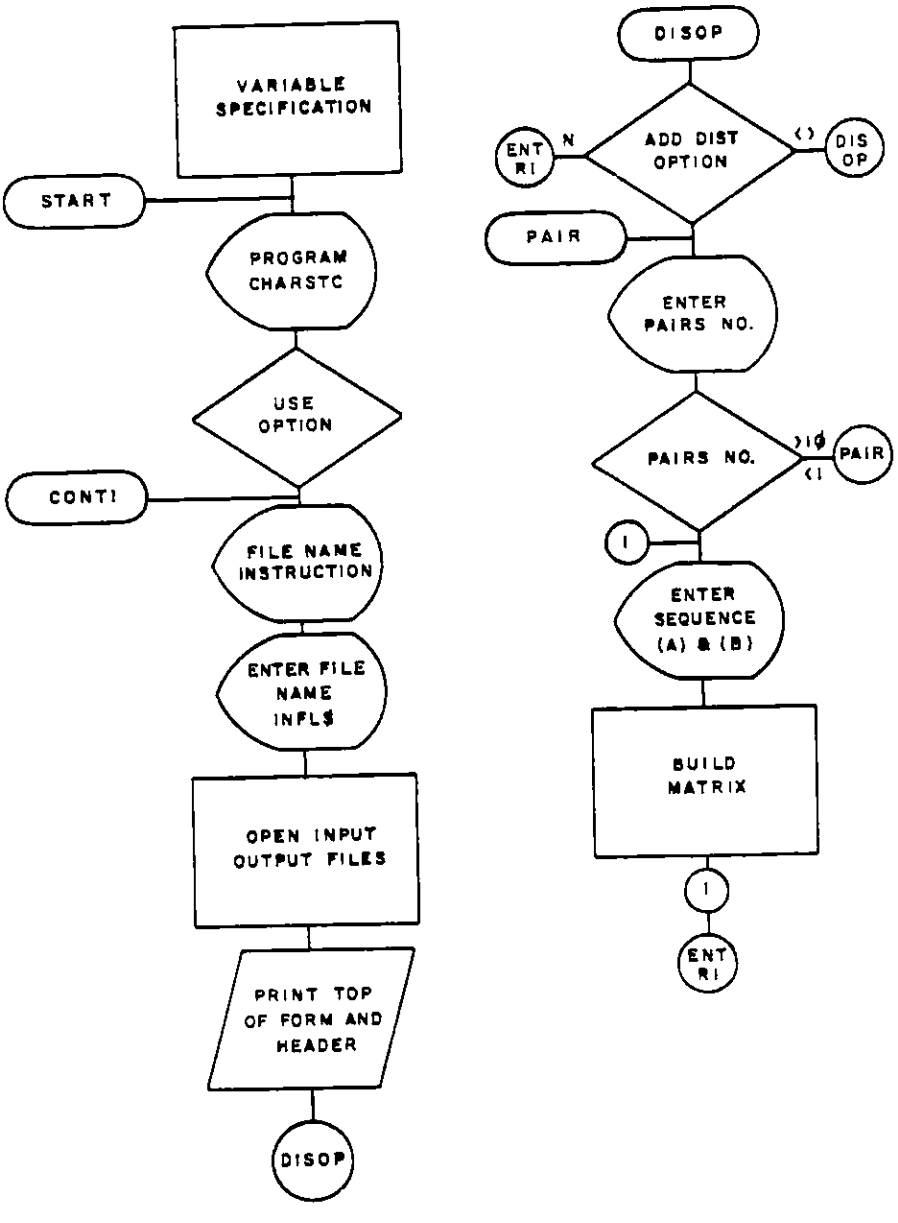
```
File DEV1:SIMDIS.BAS[1,4] printed on 0/0/0

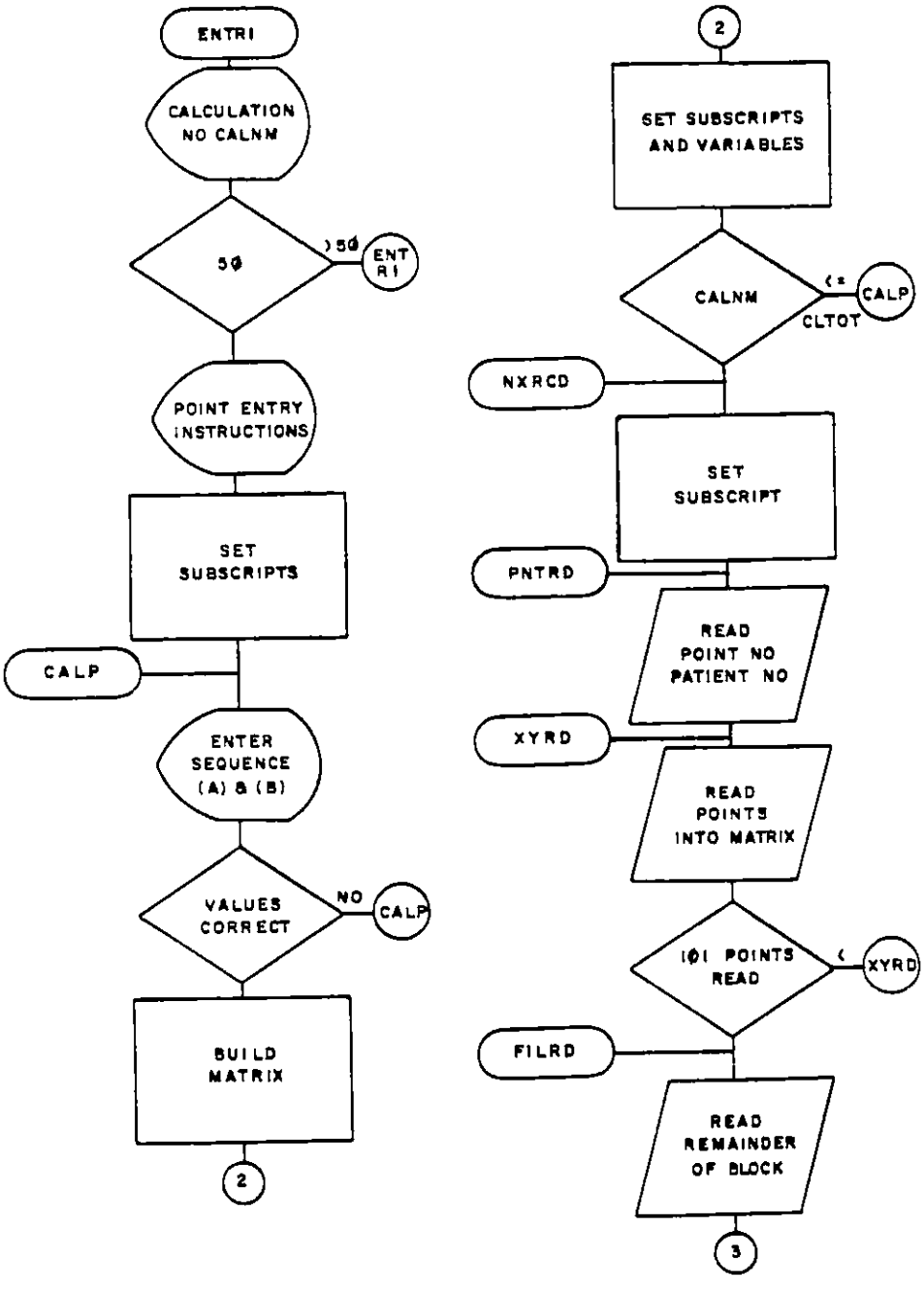
        INPUT "DO YOU WISH TO USE THE ADD DISTANCE OPTION (Y OR N): ",ANS$
        IF ANS$ = "N" GOTO ENTR1
        IF ANS$ <> "Y" GOTO DISUP
        FLG = 1
FAIR :  PRINT TAB(-1,0);TAB(15,5)
        INPUT "ENTER THE NUMBER OF PAIRS TO BE ADDED (MAX = 10): ",PRS
        IF PRS > 10 GOTO FAIR
        IF PRS < 1 GOTO FAIR
        FOR SUB1 = 1 TO PRS
        PRINT TAB(-1,0);TAB(15,5)
        INPUT "ENTER CALCULATION SEQUENCE NUMBER (A) :",CALCA
        PRINT " ";TAB(15,5)
        INPUT "ENTER CALCULATION SEQUENCE NUMBER (B) :",CALCB
        CALC2(SUB1) = CALCA
        CALC3(SUB1) = CALCB
        NEXT SUB1
'
ENTR1 : PRINT TAB(-1,0);TAB(15,5)
        INPUT "ENTER THE NUMBER OF CALCULATIONS (MAX=50) :",CALNM
        IF CALNM > 50 GOTO ENTR1
'
        PRINT TAB(-1,0)
        PRINT "TWO POINTS WILL BE NEED FOR EACH OF THE CALCULATIONS."
        PRINT "THESE WILL BE REFERED TO AS POINT (A) AND POINT (B)."
        PRINT "THE SEQUENCE NUMBERS OF EACH POINT SHOULD BE THE"
        PRINT "RESPONSE. IF IT IS DESIRED TO EXIT FROM THE PROGRAM"
        PRINT "ENTER A (0) FOR POINT (A)."
        PRINT " "
        INPUT "PRESS RETURN TO CONTINUE",ANS$
'
        LET SUB1 = 1
        LET CLTOT = 1
CALC :  PRINT TAB(-1,0);TAB(15,5)
        PRINT "THIS IS CALCULATION NUMBER : ";CLTOT;TAB(17,5)
        INPUT "ENTER POINT (A) : ",PTA
        IF PTA > 100 GOTO CALC
        IF PTA = 0 GOTO ENDRT
        IF PTA < 0 GOTO CALC
        PRINT " ";TAB(15,5)
        INPUT "ENTER POINT (B) : ",PTB
        IF PTB > 100 GOTO CALC
        IF PTB < 1 GOTO CALC
        PRINT " ";TAB(20,5)
        INPUT "ARE THE (A) AND (B) VALUES CORRECT (Y OR N) : ",COR$
        IF COR$ = "N" GOTO CALC
        LET A(SUB1) = PTA
        LET B(SUB1) = PTB
        LET CLTOT = CLTOT + 1
        LET SUB1 = SUB1 + 1
        IF CLTOT <= CALNM GOTO CALC
'
NXRCD : LET SUB1 = 1
PNTRD : INPUT #5,OFFT,PNB
XYRD :  INPUT #5,XPTR(SUB1),YPTR(SUB1)
        LET SUB1 = SUB1 + 1
        IF SUB1 < 101 GOTO  XYRD
        LET SUB1 = 1
FILRD : INPUT #5,FILL
        LET SUB1 = SUB1 + 1
        IF SUB1 < 21 GOTO FILRD
```

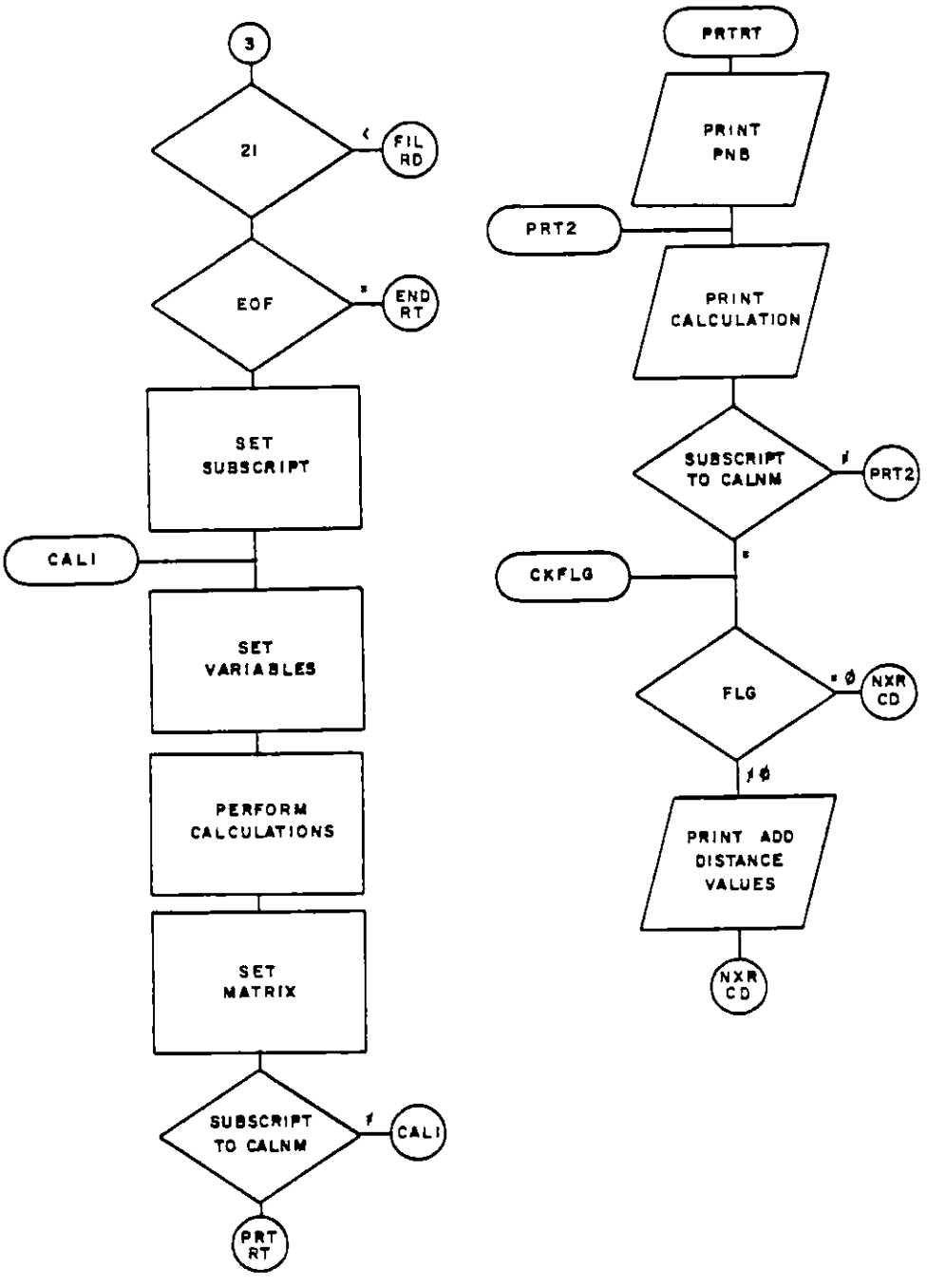
File DEH1:SIMDIS.BAS[1,4] printed on 0/0/0

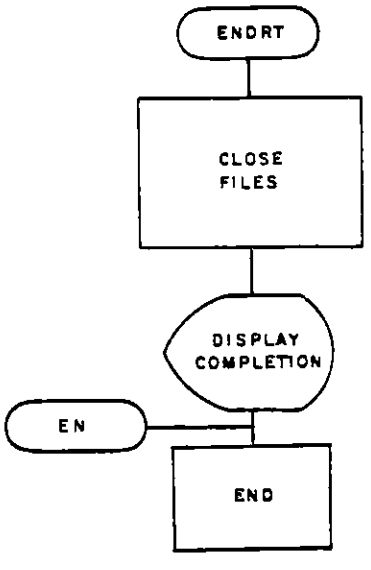
```
        IF EOF(5) = 1 GOTO ENDRT
        LET SUB1 = 1
CAL1 :  LET SUBA = A(SUB1)
        LET SUBB = B(SUB1)
        LET XASV = XSTR(SUBA)
        LET YASV = YSTR(SUBA)
        LET XBSV = XSTR(SUBB)
        LET YBSV = YSTR(SUBB)
        LET DIFSV = SQR((XASV - XBSV)**2 + (YASV - YBSV)**2)
        LET L = DIFSV/200
        LET L = L * 25.4
        LET IN(SUB1) = L
        IF SUB1 = CALNM GOTO PRTRT
        LET SUB1 = SUB1 +1
        GOTO CAL1

'
PRTRT : PRINT#7,PNB
        LET SUB1 = 1
PRT2 :  PRINT#7,TAB(10);"POINT";A(SUB1);"TO";B(SUB1);"EQUALS";IN(SUB1);" mm."
        IF SUB1 = CALNM GOTO CHFLG
        LET SUB1 = SUB1 + 1
        GOTO PRT2
CHFLG : IF FLG = 0 GOTO NXRCD
        FOR SUB1 = 1 TO PRS
        CALCX = CALC2(SUB1) : CALCY = CALC3(SUB1)
        ADD1(SUB1) = IN(CALCX) + IN(CALCY)
        NEXT SUB1
        PRINT #7," "
        FOR SUB1 = 1 TO PRS
        PRINT #7,TAB(10);"CALCULATION ";CALC2(SUB1);" + ";CALC3(SUB1);
        PRINT #7,"EQUALS ";ADD1(SUB1);" mm."
        NEXT SUB1
        GOTO NXRCD
'
ENDRT : CLOSE #5
        CLOSE #7
        PRINT TAB(-1,0);TAB(15,5)
        PRINT "THE TWO POINT DISTANCE PROGRAM IS COMPLETE";TAB(16,5,
        INPUT "PRESS RETURN FOR NEXT PROGRAM",AN2$
EN :    END
```


VARIABLE SPECIFICATION
START
PROGRAM CHARSTC
USE OPTION
CONTI
FILE NAME INSTRUCTION
ENTER FILE NAME INFL$
OPEN INPUT OUTPUT FILES
PRINT TOP OF FORM AND HEADER
DISOP
DISOP
ADD DIST OPTION
N
ENT RI
<>
DIS OP
PAIR
ENTER PAIRS NO.
PAIRS NO.
>10
<1
PAIR
1
ENTER SEQUENCE (A) & (B)
BUILD MATRIX
1
ENT RI


ENTRI
CALCULATION NO CALNM
50
>50
ENTRI
POINT ENTRY INSTRUCTIONS
SET SUBSCRIPTS
CALP
ENTER SEQUENCE (A) & (B)
VALUES CORRECT
NO
CALP
BUILD MATRIX
2
2
SET SUBSCRIPTS AND VARIABLES
CALNM
<=
CALP
CLTOT
NXRCD
SET SUBSCRIPT
PNTRD
READ POINT NO PATIENT NO
XYRD
READ POINTS INTO MATRIX
10I POINTS READ
<
XYRD
FILRD
READ REMAINDER OF BLOCK
3


3
21
<
FIL RD
EOF
=
END RT
SET SUBSCRIPT
CALI
SET VARIABLES
PERFORM CALCULATIONS
SET MATRIX
SUBSCRIPT TO CALNM
≠
CALI
PRT RT
PRTRT
PRINT PNB
PRT2
PRINT CALCULATION
SUBSCRIPT TO CALNM
≠
PRT2
CKFLG
=
FLG
= Ø
NXR CD
≠ Ø
PRINT ADD DISTANCE VALUES
NXR CD


ENDRT
CLOSE
FILES
DISPLAY
COMPLETION
EN
END

APPENDIX III

ANGL3.BAS - Program and Flowchart

```
File DSK1:ANGL3.BAS[1,4] printed on 07/07/

'          ************************************************
'          *                                              *
'          *      ANGL3.BAS - THIS PROGRAM CALCULATES A   *
'          *      THREE POINT ANGLE FROM COORDINATE VALUES*
'          *      STORED IN A DISC FILE.                  *
'          *                                              *
'          ************************************************
'
'
'
          STRSIZ 20
          MAP1 COPT
             MAP2 COPT,F,3
          MAP1 PNBR
             MAP2 PNB,F,6
          MAP1 XYSTR(100)
             MAP2 XSTR,B,2
             MAP2 YSTR,B,2
          MAP1 AB(50)
             MAP2 A,F
             MAP2 B,F
             MAP2 C,F
          MAP1 DEGRE(50)
             MAP2 DES,F
          MAP1 FILR
             MAP2 FILL,B,5
'
'
START :   PRINT TAB(-1,0)
          PRINT "THIS PROGRAM IS FOR THE CALCULATION OF OF AN ANGLE"
          PRINT "DEFINED BY THREE POINTS. THE DESIRED CALCULATIONS"
          PRINT "WILL BE PERFORMED ON ALL RECORDS OF THE FILE. A"
          PRINT "MAXIMUM OF (50) CALCULATIONS MAY BE EXECUTED"
          PRINT "PER RECORD."
          PRINT " "
          INPUT "DO YOU WISH TO USE THIS OPTION (Y OR N) :",AN1$
          IF AN1$ = "N" GOTO EN
          IF AN1$ = "Y" GOTO CONT1
          GOTO START

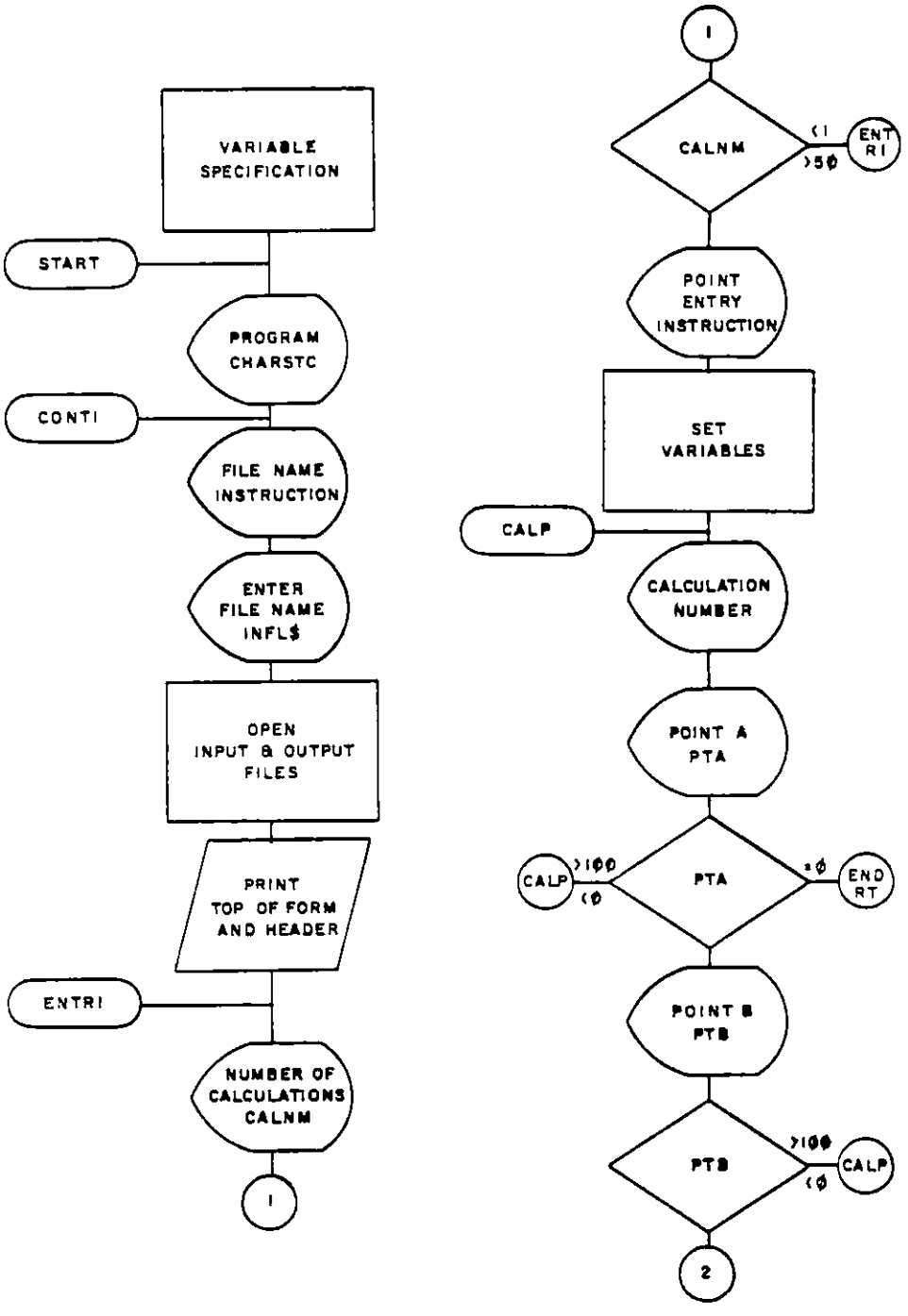
'
CONT1 :   PRINT TAB(-1,0);TAB(5,5)
          PRINT "ENTER THE NAME OF THE DATA FILE THAT IS TO BE";TAB(6,5)
          PRINT "USED. THE FILE NAME MUST BE PRECEDED BY DSK1:.";TAB(7,5)
          PRINT "THE NAME MUST BE IMMEDIATELY FOLLOWED WITH A";TAB(8,5)
          PRINT "PERIOD AND THE THREE LETTERS DAT. FOR EXAMPLE IF";TAB(9,5)
          PRINT "THE FILE NAME IS CEPH THEN ENTER DSK1:CEPH.DAT ";TAB(10,5)
          PRINT "FOR THE FILE NAME. YOUR FILE NAME WILL BE THE ";TAB(11,5)
          PRINT "SAME AS CREATED BY THE FLTLOD PROGRAM.";TAB(12,5)
          INPUT "ENTER THE FILE NAME : ",INFL$
'
          OPEN #5,INFL$,INPUT
          OPEN #7,"TRM:PRINT",OUTPUT
          PRINT #7,CHR$(12)
          PRINT #7,TAB(40);"THREE POINT ANGLE CALCULATIONS"
'
ENTR1 :   PRINT TAB(-1,0);TAB(15,5)
          INPUT "ENTER THE NUMBER OF CALCULATIONS (MAX=50) :",CALNM
          IF CALNM < 1 GOTO ENTR1
          IF CALNM > 50 GOTO ENTR1
```

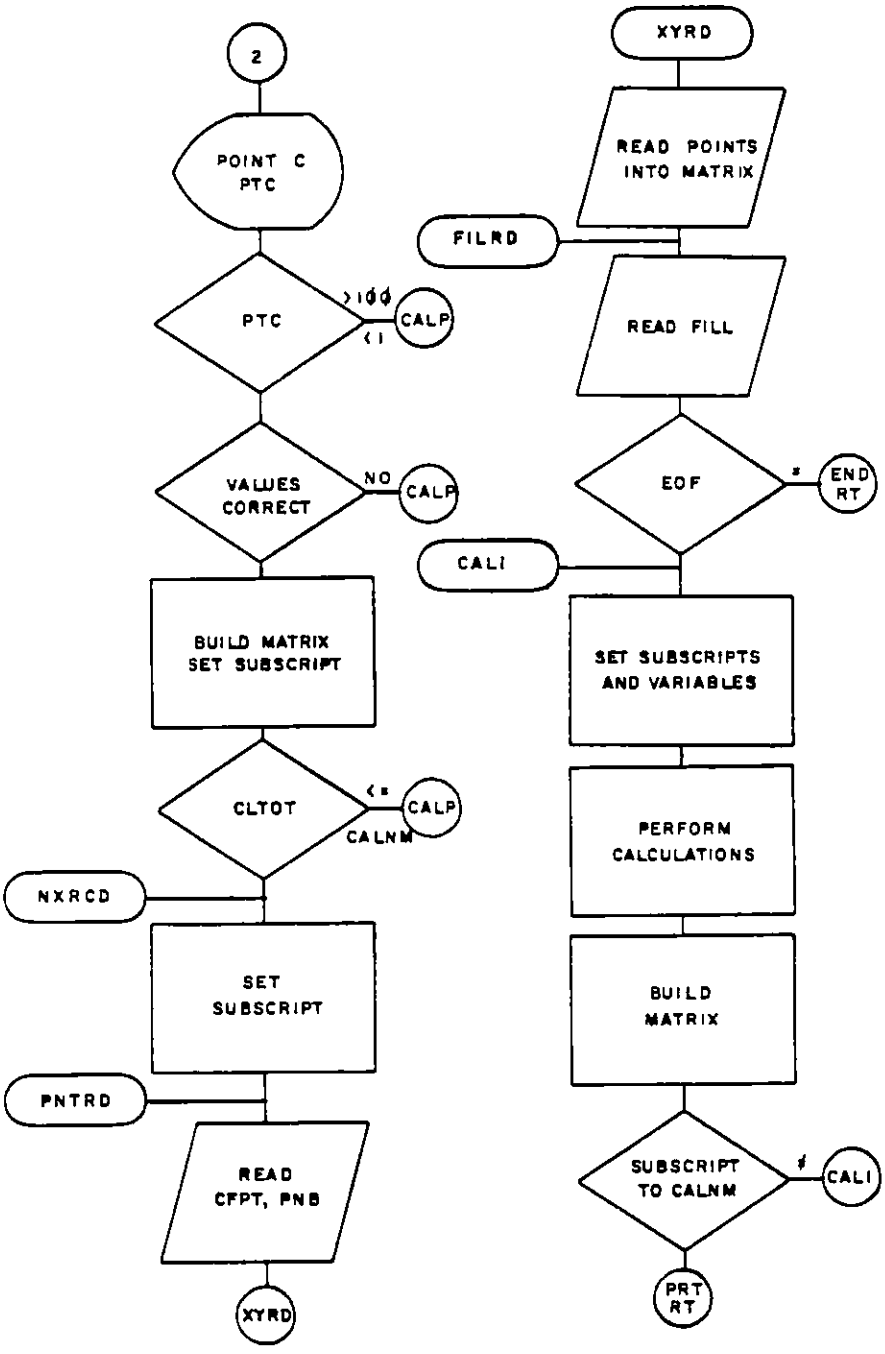
File DSK1:ANGLE.BAS[1,4] printed on 0/0/0

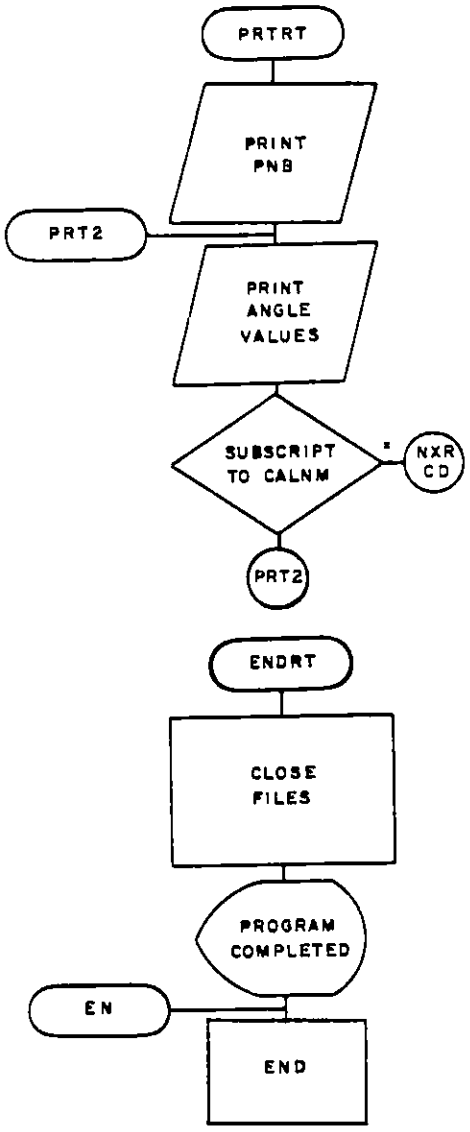
```
          PRINT TAB(-1,0)
          PRINT "THREE POINTS WILL BE ASKED FOR EACH OF THE CALCULATIONS."
          PRINT "THESE WILL BE REFERED TO AS POINT (A), POINT (B), AND"
          PRINT "POINT (C). THE SEQUENCE NUMBERS OF EACH POINT SHOULD"
          PRINT "BE THE RESPONSE. POINT (B) WILL BE CONSIDERED AS THE"
          PRINT "POINT OF THE CALCULATED ANGLE. IF IT IS DESIRED TO EXIT"
          PRINT "FROM THE PROGRAM ENTER A (0) FOR POINT (A)."
          INPUT "PRESS RETURN TO CONTINUE :",ANS$
'
          LET SUB1 = 1
          LET CLTOT = 1
CALP :    PRINT TAB(-1,0);TAB(15,5)
          PRINT "THIS IS CALCULATION NUMBER : ";CLTOT;TAB(17,5)
          INPUT "ENTER POINT (A) :",PTA
          IF PTA > 100 GOTO CALP
          IF PTA = 0 GOTO ENDRT
          IF PTA < 0 GOTO CALP
          PRINT " ";TAB(13,5)
          INPUT "ENTER POINT (B) :",PTB
          IF PTB > 100 GOTO CALP
          IF PTB < 1 GOTO CALP
          PRINT " ";TAB(13,5)
          INPUT "ENTER POINT (C) :",PTC
          IF PTC > 100 GOTO CALP
          IF PTC <1 GOTO CALP
          PRINT " ";TAB(21,5)
          INPUT "ARE THE VALUES FOR (A),(B), AND (C) CORRECT (Y OR N) : ";COR$
          IF COR$ = "N" GOTO CALP
          LET A(SUB1) = PTA
          LET B(SUB1) = PTB
          LET C(SUB1) = PTC
          LET CLTOT = CLTOT + 1
          LET SUB1 = SUB1 + 1
          IF CLTOT <= CALNM GOTO CALP
'
'
NXRTD :   LET SUB1 = 1
ENTRD :   INPUT #5,CPRT,ENB
XYRD :    INPUT #5,XPTR(SUB1),YPTR(SUB1)
          LET SUB1 = SUB1 + 1
          IF SUB1 > 101 GOTO XYRD
          LET SUB1 = 1
FILRD :   INPUT #5,FILL
          LET SUB1 = SUB1 + 1
          IF SUB1 > 21 GOTO FILRD
          IF EOF(5) = 1 GOTO ENDRT
          LET SUB1 = 1
CALC :    LET SUBA = A(SUB1)
          LET SUBB = B(SUB1)
          LET SUBC = C(SUB1)
          LET XAPV = XPTR(SUBA)
          LET YAPV = YPTR(SUBA)
          LET XBPV = XPTR(SUBB)
          LET YBPV = YPTR(SUBB)
          LET XCPV = XPTR(SUBC)
          LET YCPV = YPTR(SUBC)
          LET DIFPV1 = SQR((XAPV - XBPV)**2 + (YAPV - YBPV)**2)
          LET DIFPV2 = SQR((XBPV - XCPV)**2 + (YBPV - YCPV)**2)
          LET DIFPV3 = SQR((XCPV - XAPV)**2 + (YCPV - YAPV)**2)
          LET C = DIFPV1
```

File DSK1:ANGL3.BAS[1,4] printed on 6/6/6

```
          LET Y = DIFSV2
          LET Z = DIFSV3
          LET RAD1 = ACS(((Z*Z)-(Y*Y)-(X*X))/-(2*Y*X))
          DEG(SUB1) = (RAD1*180)/3.1417
          IF SUB1 = CALNM GOTO PRTRT
          LET SUB1 = SUB1 + 1
          GOTO CALL
'
'
PRTRT :   PRINT #7,PNB
          LET SUB1 = 1
PRT2 :     PRINT #7,TAB(10);"ANGLE OF POINT ";B(SUB1);" = ";DEG(SUB1);" DEG S"
          IF SUB1 = CALNM GOTO NXACD
          LET SUB1 = SUB1 + 1
          GOTO PRT2
'
'
ENDRT :   CLOSE #5
          CLOSE #7
          PRINT TAB(-1,0);TAB(15,5)
          PRINT "THE THREE POINT ANGLE PROGRAM IS COMPLETE.";TAB(16,5)
          INPUT "PRESS RETURN FOR NEXT PROGRAM.",ANS$
END :     END
```


VARIABLE SPECIFICATION
START
PROGRAM CHARSTC
CONTI
FILE NAME INSTRUCTION
ENTER FILE NAME INFL$
OPEN INPUT & OUTPUT FILES
PRINT TOP OF FORM AND HEADER
ENTRI
NUMBER OF CALCULATIONS CALNM
1
1
CALNM
<1
>50
ENTRI
POINT ENTRY INSTRUCTION
SET VARIABLES
CALP
CALCULATION NUMBER
POINT A PTA
CALP
>100
<0
PTA
=0
ENDRT
POINT B PTB
PTB
>100
<0
CALP
2


2
POINT C
PTC
PTC
>100
<1
CALP
VALUES
CORRECT
NO
CALP
BUILD MATRIX
SET SUBSCRIPT
CLTOT
<=
CALP
CALNM
NXRCD
SET
SUBSCRIPT
PNTRD
READ
CFPT, PNB
XYRD
XYRD
READ POINTS
INTO MATRIX
FILRD
READ FILL
EOF
=
END
RT
CALI
SET SUBSCRIPTS
AND VARIABLES
PERFORM
CALCULATIONS
BUILD
MATRIX
SUBSCRIPT
TO CALNM
≠
CALI
PRT
RT


PRTRT
PRINT
PNB
PRT2
PRINT
ANGLE
VALUES
SUBSCRIPT
TO CALNM
NXR
CD
PRT2
ENDRT
CLOSE
FILES
PROGRAM
COMPLETED
EN
END

APPENDIX IV

ANGL4.BAS - Program and Flowchart

File DSK1:ANGL4.BAS[1,4] printed on 07/07/0

```
'       ************************************************************
'       *                                                          *
'       *   ANGL4.BAS - THIS PROGRAM CALCULATES AN ANGLE           *
'       *   DEFINED BY FOUR POINTS FROM COORDINATE VALUES          *
'       *   STORED IN A DISC FILE.                                 *
'       *                                                          *
'       ************************************************************
'
'
        STRSIZ 20
        MAP1 CPPT
           MAP2 CPT.F,2
        MAP1 PNBR
           MAP2 PNB.F
        MAP1 XYSTR(100)
           MAP2 XSTR,B,2
           MAP2 YSTR,B,2
        MAP1 AB(50)
           MAP2 A,F
           MAP2 B,F
           MAP2 C,F
           MAP2 D,F
        MAP1 DEGRE(50)
           MAP2 DEG,F
        MAP1 FILR
           MAP2 FILL,3,5
'
'
START :  PRINT TAB(-1,0)
         PRINT "THIS PROGRAM IS FOR THE CALCULATION OF AN ANGLE"
         PRINT "DEFINED BY FOUR POINTS. THE DESIRED CALCULATIONS"
         PRINT "WILL BE PERFORMED ON ALL RECORDS OF THE FILE. A "
         PRINT "MAXIMUM OF (50) CALCULATIONS MAY BE EXECUTED "
         PRINT "PER RECORD."
         PRINT " "
         INPUT "DO YOU WISH TO USE THIS OPTION (Y OR N) :",ANI$
         IF ANI$ = "N" GOTO EN
         IF ANI$ = "Y" GOTO CONT1
         GOTO START
'
CONT1 :  PRINT TAB(-1,0);TAB(5,5)
         PRINT "ENTER THE NAME OF THE DATA FILE THAT IS TO BE";TAB(5,5)
         PRINT "USED. THE FILE NAME MUST BE PRECEDED BY DSK1:.";TAB(7,5)
         PRINT "THE NAME MUST BE IMMEDIATELY FOLLOWED WITH A";TAB(8,5)
         PRINT "PERIOD AND THE THREE LETTERS DAT. FOR EXAMPLE IF";TAB(9,5)
         PRINT "THE FILE NAME IS CEPH THEN ENTER DSK1:CEPH.DAT";TAB(10,5)
         PRINT "FOR THE FILE NAME. YOUR FILE NAME WILL BE THE";TAB(11,5)
         PRINT "SAME AS CREATED BY THE FLTLOD PROGRAM.";TAB(12,5)
         INPUT "ENTER THE FILE NAME : ",INFL$
'
         OPEN #5,INFL$,INPUT
         OPEN #7,"TRM:PRINT",OUTPUT
         PRINT #7,CHR$(12)
         PRINT #7,TAB(40);"FOUR POINT ANGLE CALCULATIONS"
'
ENTR1 :  PRINT TAB(-1,0);TAB(15,5)
         INPUT "ENTER THE NUMBER OF CALCULATIONS (MAX=50) :",CALNM
         IF CALNM < 1 GOTO ENTR1
         IF CALNM > 50 GOTO ENTR1
         PRINT TAB(-1,0)
```

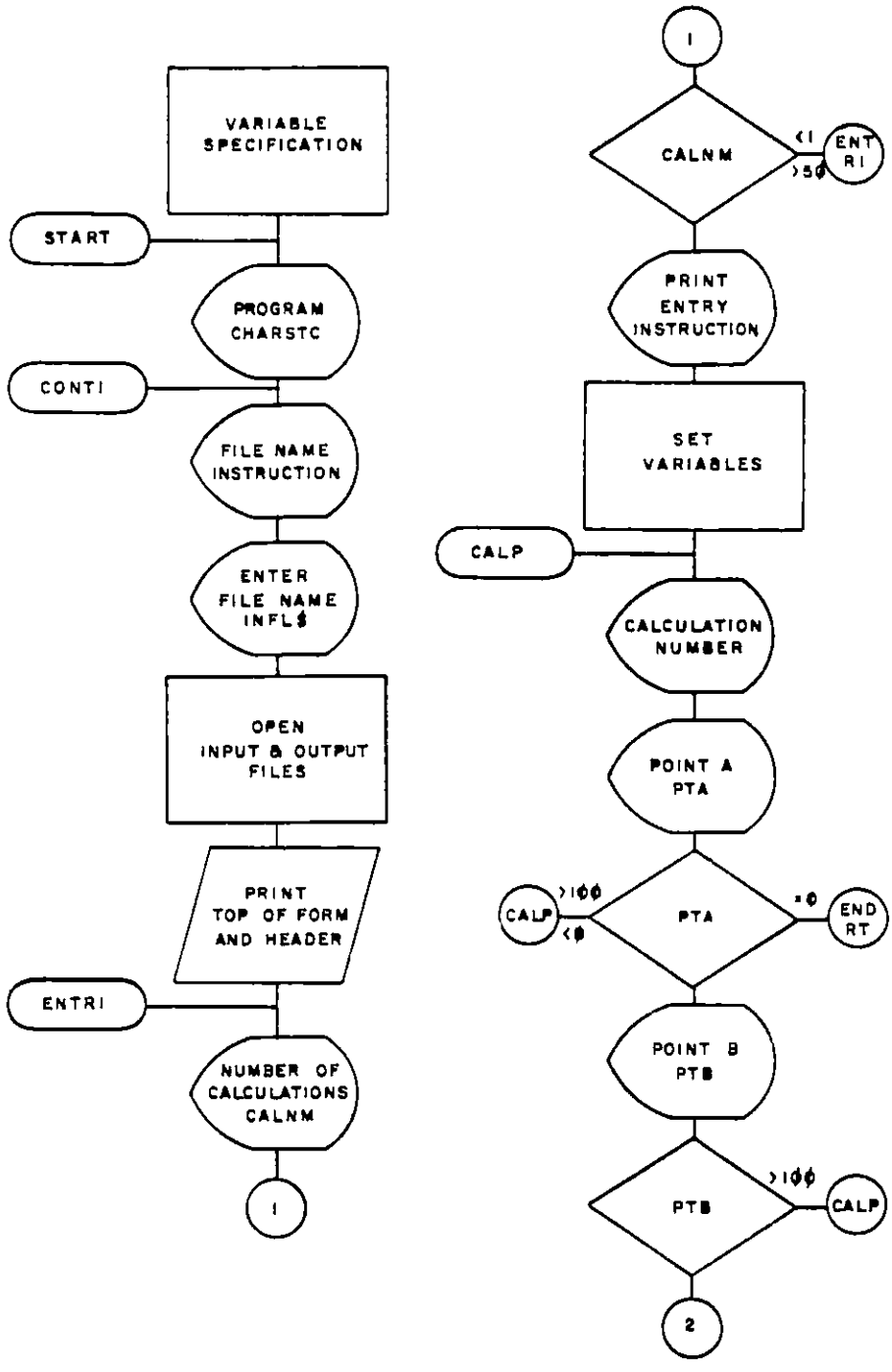
```
File DB1:ANGL4.BAS[1,4] printed on 0/0/0

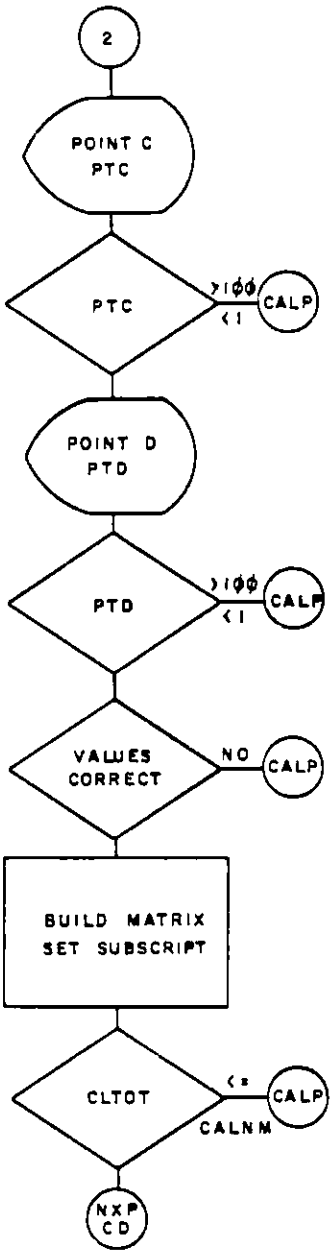
          PRINT "FOUR POINTS WILL BE NEED FOR EACH OF THE CALCULATIONS."
          PRINT "THESE WILL BE REFERED TO AS POINT (A), POINT (B), POINT"
          PRINT "(C), AND POINT (D). THE SEQUENCE NUMBERS OF EACH POINT"
          PRINT "SHOULD BE THE RESPONSE. POINT (A) AND (C) WILL DEFINE THE"
          PRINT "THE CALCULATED ANGLE. IF IT IS DESIRED TO EXIT FROM THE"
          PRINT "PROGRAM ENTER A (0) FOR POINT (A)."
          INPUT "PRESS RETURN TO CONTINUE :",ANS$

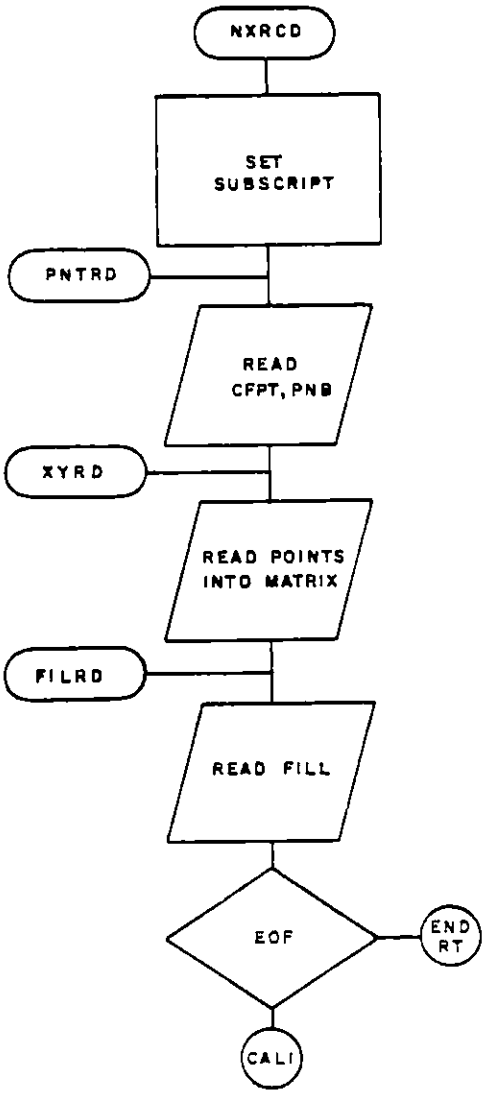
          LET SUB1 = 1 : LET CLTOT = 1
CALP :    PRINT TAB(-1,0);TAB(15,5)
          PRINT "THIS IS CALCULATION NUMBER : ";CLTOT;TAB(17,5)
          INPUT "ENTER POINT (A) :",PTA
          IF PTA > 100 GOTO CALP
          IF PTA = 0 GOTO ENDRT
          IF PTA < 0 GOTO CALP
          PRINT " ";TAB(18,5)
          INPUT "ENTER POINT (B) :",PTB
          IF PTB > 100 GOTO CALP
          IF PTB < 1 GOTO CALP
          PRINT " ";TAB(19,5)
          INPUT "ENTER POINT (C) :",PTC
          IF PTC > 100 GOTO CALP
          IF PTC < 1 GOTO CALP
          PRINT " ";TAB(20,5)
          INPUT "ENTER POINT (D) :",PTD
          IF PTD > 100 GOTO CALP
          IF PTD < 1 GOTO CALP
          PRINT " ";TAB(22,5)
          INPUT "ARE ALL VALUES CORRECT (Y OR N) : ",COR$
          IF COR$ = "N" GOTO CALP
          LET A(SUB1) = PTA
          LET B(SUB1) = PTB
          LET C(SUB1) = PTC
          LET D(SUB1) = PTD
          LET CLTOT = CLTOT + 1
          LET SUB1 = SUB1 + 1
          IF CLTOT <= CALNM GOTO CALP
'
'
NXRCD :   LET SUB1 = 1
ENTRD :   INPUT #5,CPPT,PNB
XYRD :    INPUT #5,XSTR(SUB1),YSTR(SUB1)
          LET SUB1 = SUB1 + 1
          IF SUB1 > 101 GOTO XYRD
          LET SUB1 = 1
FILRD :   INPUT #5,FILL
          LET SUB1 = SUB1 + 1
          IF SUB1 < 21 GOTO FILRD
          IF EOF(5) = 1 GOTO ENDRT
          LET SUB1 = 1
'
CALC :    LET SUBA = A(SUB1) : LET SUBB =B(SUB1)
          LET SUBC = C(SUB1) : LET SUBD =D(SUB1)
          LET XAEV = XSTR(SUBA) : LET YAEV = YSTR(SUBA)
          LET XBEV = XSTR(SUBB) : LET YBEV = YSTR(SUBB)
          LET XCEV = XSTR(SUBC) : LET YCEV = YSTR(SUBC)
          LET XDEV = XSTR(SUBD) : LET YDEV = YSTR(SUBD)
          LET TXEV = XCEV - XAEV
          LET TYEV = YCEV - YAEV
          LET XDEV = XDEV - TXEV
```

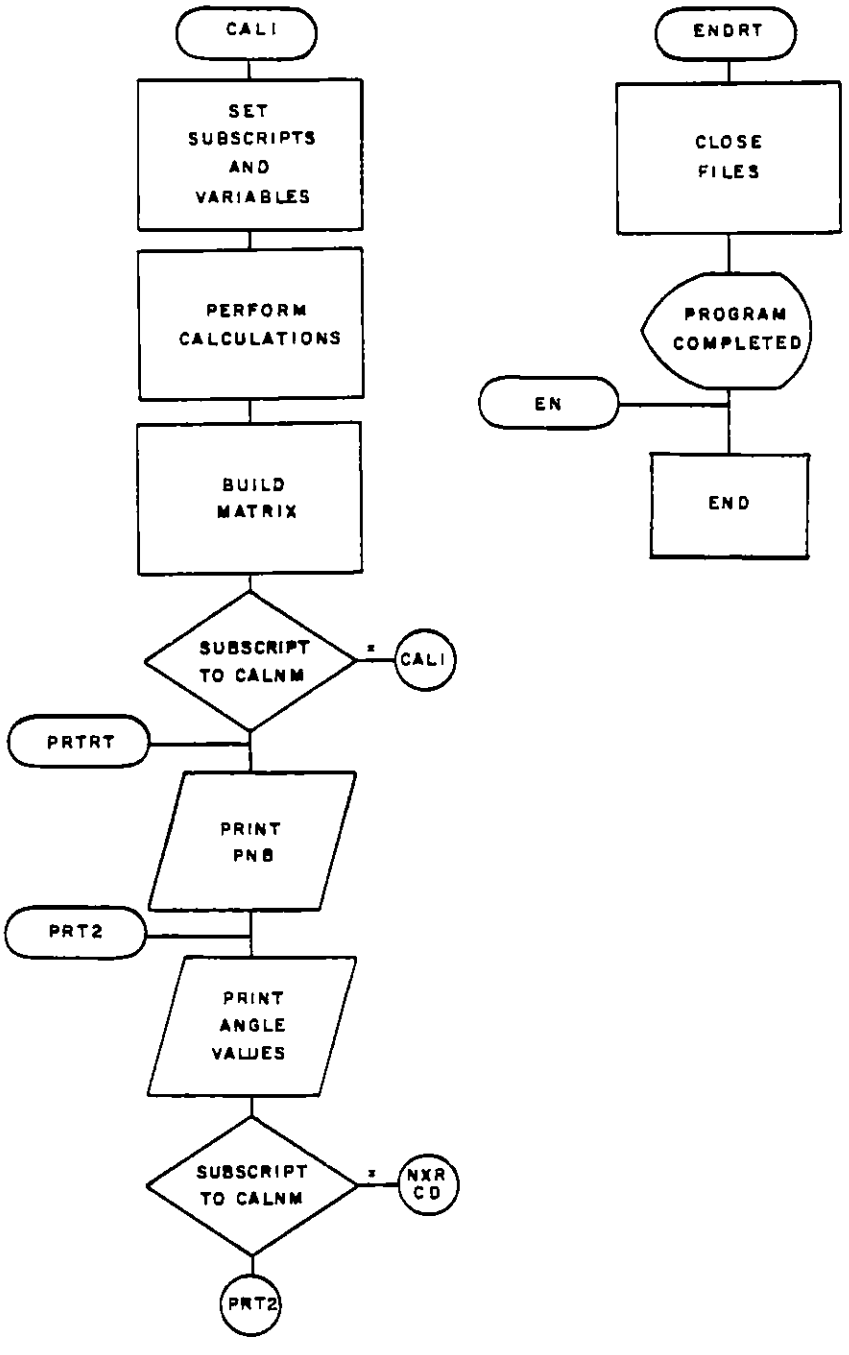
```
File DB1:ANGL4.BAS[1,4] printed on 6/6/6

          LET YDBV = YDBV - TYBV
          LET XCBV = XABV
          LET YCBV = YABV
          LET DIFBV1 = SQR((XCBV-XBBV)**2+(YCBV-YBBV)**2)
          LET DIFBV2 = SQR((XBBV-XDBV)**2+(YBBV-YDBV)**2)
          LET DIFBV3 = SQR((XDBV-XCBV)**2+(YDBV-YCBV)**2)
          LET X = DIFBV1
          LET Y = DIFBV2
          LET Z = DIFBV3
          LET RAD1 = ACS(((Y*Y)-(X*X)-(Z*Z))/-(2*Z*X))
          DEG(SUB1) = (RAD1*180)/3.1417
          IF SUB1 = CALNM GOTO PRTRT
          SUB1 = SUB1 + 1
          GOTO CAL1
!
!
PRTRT :   PRINT #7,PNB
          SUB1 = 1
PRTI :    PRINT #7,TAB(10);"ANGLE VALUE =  ";DEG(SUB1);" DEG F"
          IF SUB1 = CALNM GOTO NXRCD
          SUB1 = SUB1 + 1
          GOTO PRTI
!
!
ENDRT :   CLOSE #5
          CLOSE #7
          PRINT TAB(-1,0);TAB(15,5)
          PRINT "THE FOUR POINT ANGLE PROGRAM IS COMPLETE.";TAB(14,5)
          INPUT "PRESS RETURN FOR NEXT PROGRAM.",ANS$
EN :      END
```


VARIABLE SPECIFICATION
START
PROGRAM CHARSTC
CONTI
FILE NAME INSTRUCTION
ENTER FILE NAME INFL$
OPEN INPUT & OUTPUT FILES
PRINT TOP OF FORM AND HEADER
ENTRI
NUMBER OF CALCULATIONS CALNM
1
1
CALNM
<1
>50
ENTRI
PRINT ENTRY INSTRUCTION
SET VARIABLES
CALP
CALCULATION NUMBER
POINT A PTA
CALP
>100
<0
PTA
=0
ENDRT
POINT B PTB
PTB
>100
CALP
2


2
POINT C
PTC
PTC
>100
<1
CALP
POINT D
PTD
PTD
>100
<1
CALP
VALUES
CORRECT
NO
CALP
BUILD MATRIX
SET SUBSCRIPT
CLTOT
<2
CALP
CALNM
NXP
CD


NXRCD
SET
SUBSCRIPT
PNTRD
READ
CFPT, PNB
XYRD
READ POINTS
INTO MATRIX
FILRD
READ FILL
EOF
END
RT
CALI


CALI
SET SUBSCRIPTS AND VARIABLES
PERFORM CALCULATIONS
BUILD MATRIX
SUBSCRIPT TO CALNM
=
CALI
PRTRT
PRINT PNB
PRT2
PRINT ANGLE VALUES
SUBSCRIPT TO CALNM
=
NXR CD
PRT2
ENDRT
CLOSE FILES
PROGRAM COMPLETED
EN
END

APPENDIX V

PTLIN.BAS - Program and Flowchart

File DSK1:FTLIN.BAS[1,4] printed on 0/0/0

```
'       ************************************************
'       *                                              *
'       *    FTLIN.BAS - THIS PROGRAM CALCULATES A     *
'       *    PERPENDICULAR DISTANCE FROM A POINT TO A  *
'       *    LINE.                                     *
'       *                                              *
'       ************************************************
'
'
        STRSIZ 20
        MAP1 CPFT
           MAP2 CPFT,F,6
        MAP1 PNBR
           MAP2 PNB,F,6
        MAP1 XYSTR(100)
           MAP2 XSTR,B,2
           MAP2 YSTR,B,2
        MAP1 AB(50)
           MAP2 A,F
           MAP2 B,F
           MAP2 C,F
        MAP1 FILR
           MAP2 FILL,S,5
        MAP1 LIN(50)
           MAP2 IN,F,6
'
'
START :  PRINT TAB(-1,0)
         PRINT "THIS PROGRAM IS FOR THE CALCULATION OF A PERPENDICULAR"
         PRINT "DISTANCE FROM A POINT TO A LINE. THE DESIRED"
         PRINT "CALCULATIONS WILL BE PERFORMED ON ALL RECORDS OF THE"
         PRINT "FILE. A MAXIMUM OF (50) CALCULATIONS MAY BE"
         PRINT "EXECUTED PER RECORD."
         PRINT " "
         INPUT "DO YOU WISH TO USE THIS PROGRAM (Y OR N) :",AN1$
         IF AN1$ = "N" GOTO EN
         IF AN1$ = "Y" GOTO CONT1
         GOTO START
'
CONT1 :  PRINT TAB(-1,0);TAB(5,5)
         PRINT "ENTER THE NAME OF THE DATA FILE THAT IS TO BE USED.";TAB(5,5)
         PRINT "THE FILE NAME MUST BE PRECEDED BY DSK1.";TAB(7,5)
         PRINT "THE NAME MUST BE IMMEDIATELY FOLLOWED WITH";TAB(9,5)
         PRINT "A PERIOD AND THE THREE LETTERS INT. FOR EXAMPLE";TAB(7,5)
         PRINT "IF THE FILE NAME IS CEPH ONE SHOULD ENTER ";TAB(10,5)
         PRINT "DSK1:CEPH.DAT FOR THE FILE NAME. YOUR FILE";TAB(11,5)
         PRINT "NAME WILL BE THE SAME AS CREATED BY THE";TAB(12,5)
         PRINT "FLTLOD PROGRAM.";TAB(13,5)
         INPUT "ENTER THE FILE NAME : ",INFL$
'
         OPEN #5,INFL$,INPUT
         OPEN #7,"TRM:PRINT",OUTPUT
         PRINT #7,CHR(12)
         PRINT #7,TAB(40);"PERPENDICULAR DISTANCE CALCULATIONS"
'
ENTR1 :  PRINT TAB(-1,0);TAB(10,5)
         INPUT "ENTER THE NUMBER OF CALCULATIONS (MAX = 50) :",CALNM
         IF CALNM < 1 GOTO ENTR1
         IF CALNM > 50 GOTO ENTR1
         PRINT TAB(-1,0)
```

```
File DF1:FTLIN.BAS[1,4] printed on 6/20/0

          PRINT "THREE POINTS WILL BE NEED FOR EACH OF THE CALCULATIONS."
          PRINT "THESE WILL BE REFERED TO AS POINT (A), POINT (B), AND"
          PRINT "POINT (C). THE SEQUENCE NUMBERS OF EACH POINT SHOULD"
          PRINT "BE THE  RESPONSE. POINTS (A), AND (B) WILL DEFINE THE "
          PRINT "LINE AND WILL BE USED FOR FIRST. POINT (C) WILL BE"
          PRINT "CONSIDERED THE POINT FROM WHICH THE PERPENDICULAR"
          PRINT "IS TO BE CALCULATED. IF IT IS DESIRED TO EXIT FROM THE"
          PRINT "PROGRAM ENTER A (0) FOR POINT (A)."
          PRINT " "
          INPUT "PRESS RETURN TO CONTINUE :",ANS$

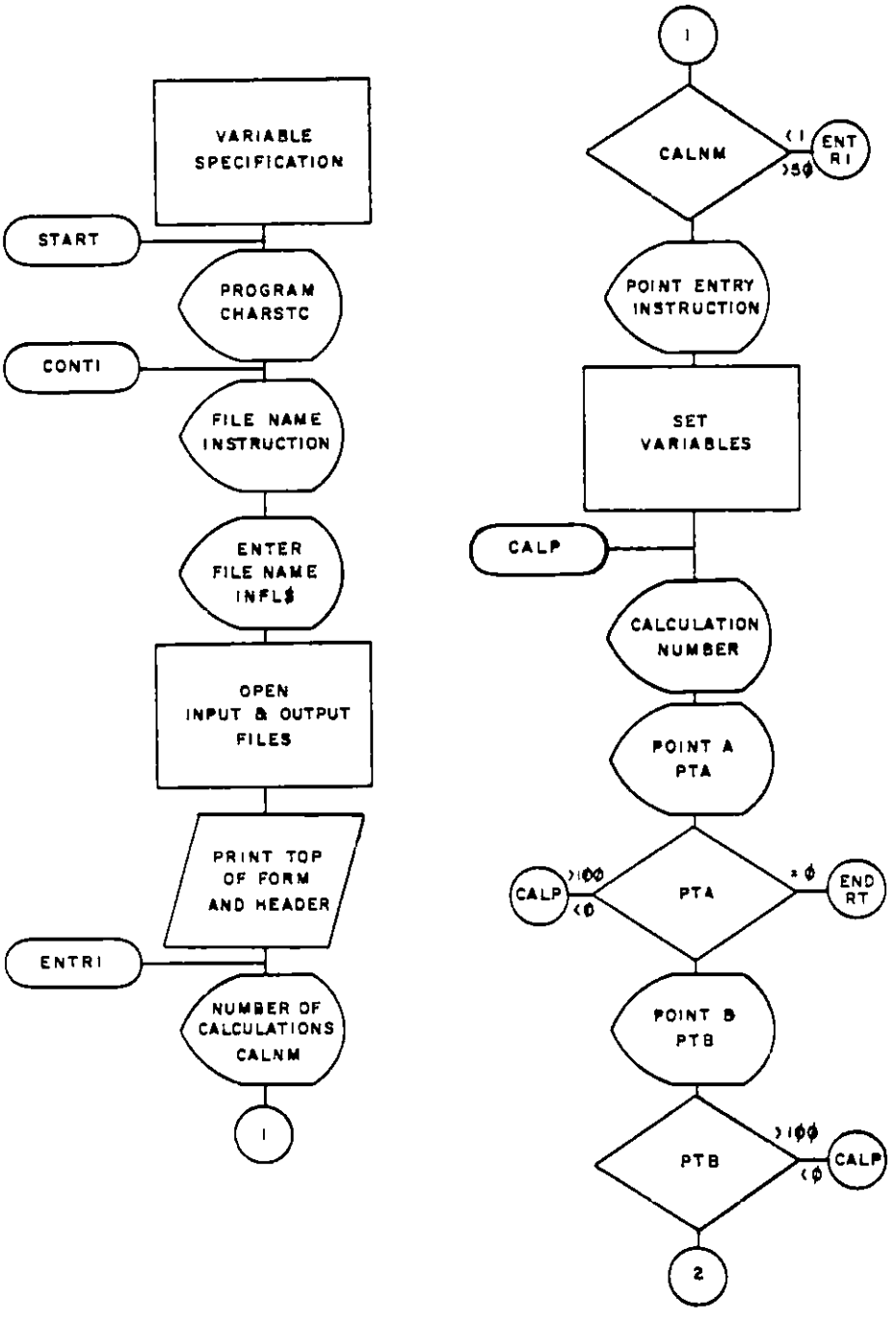
          SUB1 = 1
          CLTOT = 1

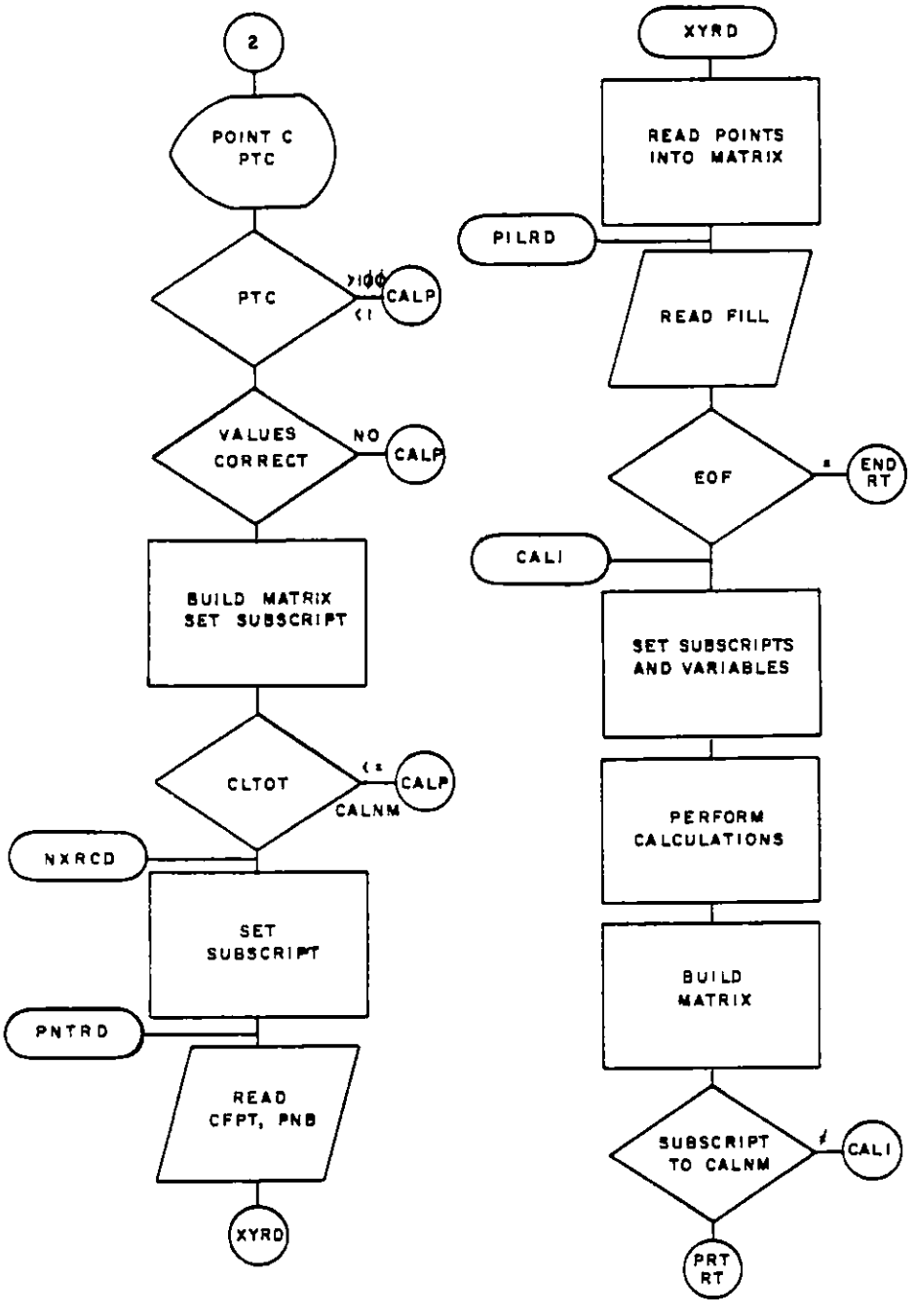
CALP :    PRINT TAB(-1,0);TAB(15,5)
          PRINT "THIS IS CALCULATION NUMBER : ";CLTOT;TAB(17,5)
          INPUT "ENTER POINT (A) :",PTA
          IF PTA  100 GOTO CALP
          IF PTA = 0 GOTO ENDRT
          IF PTA  0 GOTO CALP
          PRINT " ";TAB(13,5)
          INPUT "ENTER POINT (B) :",PTB
          IF PTB  100 GOTO CALP
          IF PTB  1 GOTO CALP
          PRINT " ";TAB(19,5)
          INPUT "ENTER POINT (C) :",PTC
          IF PTC  100 GOTO CALP
          IF PTC  1 GOTO CALP
          PRINT " ";TAB(21,5)
          INPUT "ARE ALL VALUES CORRECT : ",COR$
          IF COR$ = "N" GOTO CALP
          A(SUB1) = PTA
          B(SUB1) = PTB
          C(SUB1) = PTC
          CLTOT = CLTOT + 1
          SUB1 = SUB1 +1
          IF CLTOT  = CALNM GOTO CALP

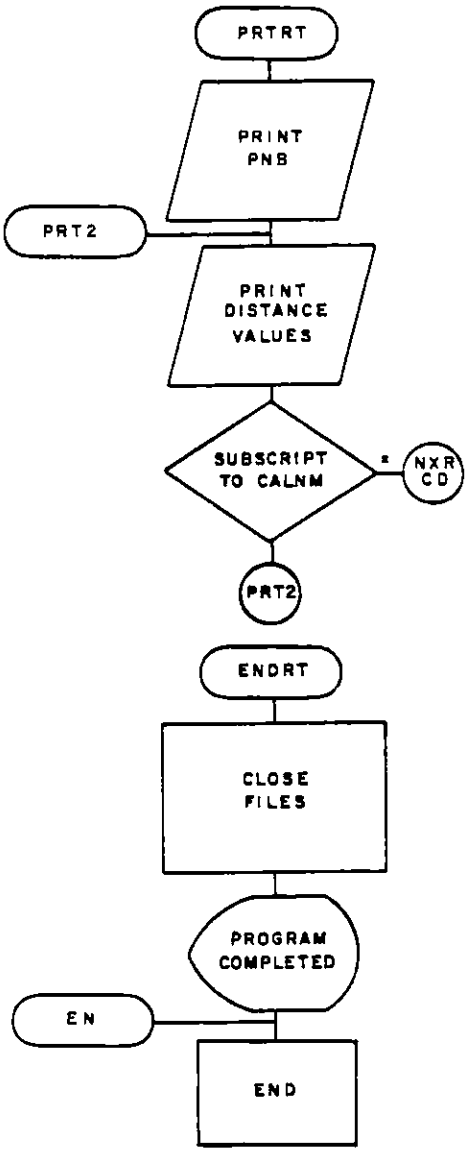
'
'
NXRCD :   SUB1 = 1
ENTRD :   INPUT #5,CHRT,PNB
XYRD :    INPUT #5,XSTR(SUB1),YSTR(SUB1)
          SUB1 = SUB1 + 1
          IF SUB1   101 GOTO XYRD
          SUB1 = 1
FILRD :   INPUT #5,FILL
          SUB1 = SUB1 +1
          IF SUB1  21 GOTO FILRD
          IF EOF(5) = 1 GOTO ENDRT
          SUB1 = 1
CALI :    SUBA = A(SUB1)
          SUBB = B(SUB1)
          SUBC = C(SUB1)
          XAEV = XSTR(SUBA)
          YAEV = YSTR(SUBA)
          XBEV = XSTR(SUBB)
          YBEV = YSTR(SUBB)
          XCEV = XSTR(SUBC)
          YCEV = YSTR(SUBC)
          DIFEV1 = SQR( (XAEV-XBEV)**2+(YAEV-YBEV)**2)
```

```
File DSK1:PTLIN.BAS[1,4] printed on 0/0/0

          DIFSV2 = SQR((xBSV-(CSV)**2+(yBSV-yCSV)**2)
          DIFSV3 = SQR((XCSV-XASV)**2+(YLSV-YASV)**2)
          X = DIFSV1
          Y = DIFSV2
          Z = DIFSV3
          RAD1 = ACS(((Y*Y)-(Z*Z)-(X*X))/-(2*Z*X))
          H = Z*SIN(RAD1)
          L = H/2.0
          L = L * 25.4
          IN(SUB1) = L
          IF SUB1 = CALNM GOTO PRTRT
          SUB1 = SUB1 + 1
          GOTO CAL1
'
PRTRT :   PRINT #7,PNB
          SUB1 = 1
PRT2 :    PRINT #7,TAB(10);"DIST.";C(SUB1);"TO LINE";A(SUB1);B(SUB1);"EQUALS";
          PRINT #7,IN(SUB1);" mm."
          IF SUB1 = CALNM GOTO NXROD
          SUB1 = SUB1 + 1
          GOTO PRT2
'
'
ENDRT :   CLOSE #5
          CLOSE #7
          PRINT TAB(-1,0);TAB(15,5)
          PRINT "THE PERPENDICULAR DISTANCE PROGRAM IS COMPLETE.";TAB(16,5)
          INPUT "PRESS RETURN FOR NEXT PROGRAM.",ANS$
EN :      END
```


VARIABLE SPECIFICATION
START
PROGRAM CHARSTC
CONTI
FILE NAME INSTRUCTION
ENTER FILE NAME INFL$
OPEN INPUT & OUTPUT FILES
PRINT TOP OF FORM AND HEADER
ENTRI
NUMBER OF CALCULATIONS CALNM
1
1
CALNM
<1
>50
ENTRI
POINT ENTRY INSTRUCTION
SET VARIABLES
CALP
CALCULATION NUMBER
POINT A PTA
CALP
>100
<0
PTA
=0
ENDRT
POINT B PTB
PTB
>100
<0
CALP
2


2
POINT C
PTC
PTC
>100
<1
CALP
VALUES
CORRECT
NO
CALP
BUILD MATRIX
SET SUBSCRIPT
CLTOT
<=
CALNM
CALP
NXRCD
SET
SUBSCRIPT
PNTRD
READ
CFPT, PNB
XYRD
XYRD
READ POINTS
INTO MATRIX
PILRD
READ FILL
EOF
=
END
RT
CALI
SET SUBSCRIPTS
AND VARIABLES
PERFORM
CALCULATIONS
BUILD
MATRIX
SUBSCRIPT
TO CALNM
≠
CALI
PRT
RT


PRTRT
PRINT
PNB
PRT2
PRINT
DISTANCE
VALUES
SUBSCRIPT
TO CALNM
=
NXR
CD
PRT2
ENDRT
CLOSE
FILES
PROGRAM
COMPLETED
EN
END

APPENDIX VI

MULDST.BAS - Program and Flowchart

File DSK1:MULDST.BAS[1,4] printed on 0/0/0

```
!          ************************************************
!          *                                              *
!          *   MULDST.BAS - THIS PROGRAM WILL CALCULATE   *
!          *   A DISTANCE BETWEEN TWO POINTS AND MULTIPLY *
!          *   THIS DISTANCE BY A VALUE.                  *
!          *                                              *
!          ************************************************
!
         STRSIZ 20
         MAP1 FNBR
            MAP2 FNB,F,6
         MAP1 XYSTR(100)
            MAP2 XSTR,F,2
            MAP2 YSTR,F,2
         MAP1 AB(50)
            MAP2 A,F
            MAP2 B,F
         MAP1 LIN(50)
            MAP2 IN,F,6
         MAP1 COEFT
            MAP2 COFT,F,4
         MAP1 FILR
            MAP2 FILL,S,5
!
!
START :  PRINT TAB(-1,0)
         PRINT "THIS PROGRAM IS FOR THE CALCULATION OF A DISTANCE BETWEEN"
         PRINT "TWO POINTS. THE CALCULATIONS ARE THEN MULTIPLIED BY A VALUE"
         PRINT "SUPPLIED BY THE OPERATOR. A MAXIMUM OF (50) CALCULATIONS"
         PRINT "MAY BE EXECUTED PER RECORD."
         PRINT " "
         INPUT "DO YOU WISH TO USE THIS OPTION (Y OR N) :",AN1$
         IF AN1$ = "N" GOTO EN
         IF AN1$ = "Y" GOTO CONT1
         GOTO START
!
CONT1 :  PRINT TAB(-1,0);TAB(5,5)
         PRINT "ENTER THE NAME OF THE DATA FILE THAT IS TO BE USED.";TAB(6,5)
         PRINT "THE FILE NAME MUST BE PRECEDED BY DSK1:.";TAB(7,5)
         PRINT "THE NAME MUST BE IMMEDIATELY FOLLOWED WITH";TAB(8,5)
         PRINT "A PERIOD AND THE THREE LETTERS DAT. FOR EXAMPLE";TAB(9,5)
         PRINT "IF THE FILE NAME IS CEFH ONE SHOULD ENTER";TAB(10,5)
         PRINT "DSK1:CEFH.DAT FOR THE FILE NAME. YOUR FILE";TAB(11,5)
         PRINT "NAME WILL BE THE SAME AS CREATED BY THE";TAB(12,5)
         PRINT "FLTLOD PROGRAM.";TAB(13,5)
         INPUT "ENTER THE FILE NAME : ",INFL$
!
         OPEN #5,INFL$,INPUT
         OPEN #7,"TRM:PRINT",OUTPUT
         PRINT #7,CHR(12)
         PRINT #7,TAB(40);"VALUE TIMES DISTANCE CALCULATIONS"
!
ENTR1 :  PRINT TAB(-1,0);TAB(15,5)
         INPUT "ENTER THE NUMBER OF CALCULATIONS (MAX=50) :",CALNM
         IF CALNM > 50 GOTO ENTR1
!
         PRINT TAB(-1,0)
         PRINT "TWO POINTS WILL BE ASKED FOR EACH OF THE CALCULATIONS."
         PRINT "THESE WILL BE REFERED TO AS POINT (A) AND POINT (B)."
```

```
File DSK1:MULDST.BAS[1,4] printed on 0/0/0

        PRINT "THE SEQUENCE NUMBERS OF EACH POINT SHOULD BE THE"
        PRINT "RESPONSE. IF IT IS DESIRED TO EXIT FROM THE PROGRAM"
        PRINT "ENTER A (0) FOR POINT (A), VALUE (X) WILL BE THE "
        PRINT "VALUE THAT EACH DISTANCE IS MULTIPLIED BY."
        PRINT " "
        INPUT "PRESS RETURN TO CONTINUE",AN2$
        PRINT TAB(-1,0);TAB(15,5)
        INPUT "ENTER VALUE (X) : "; VALX

        LET SUB1 = 1
        LET CLTOT = 1
CALP :  PRINT TAB(-1,0);TAB(15,5)
        PRINT "THIS IS CALCULATION NUMBER : ";CLTOT;TAB(17,5)
        INPUT "ENTER POINT (A) : ",PTA
        IF PTA > 100 GOTO CALP
        IF PTA = 0 GOTO ENDRT
        IF PTA < 0 GOTO CALP
        PRINT " ";TAB(15,5)
        INPUT "ENTER POINT (B) : ",PTB
        IF PTB > 100 GOTO CALP
        IF PTB < 1 GOTO CALP
        PRINT " ";TAB(20,5)
        INPUT "ARE ALL VALUES CORRECT : "; COR$
        IF COR$ = "N" GOTO CALP
        LET A(SUB1) = PTA
        LET B(SUB1) = PTB
        LET CLTOT = CLTOT + 1
        LET SUB1 = SUB1 + 1
        IF CLTOT <= CALNM GOTO CALP

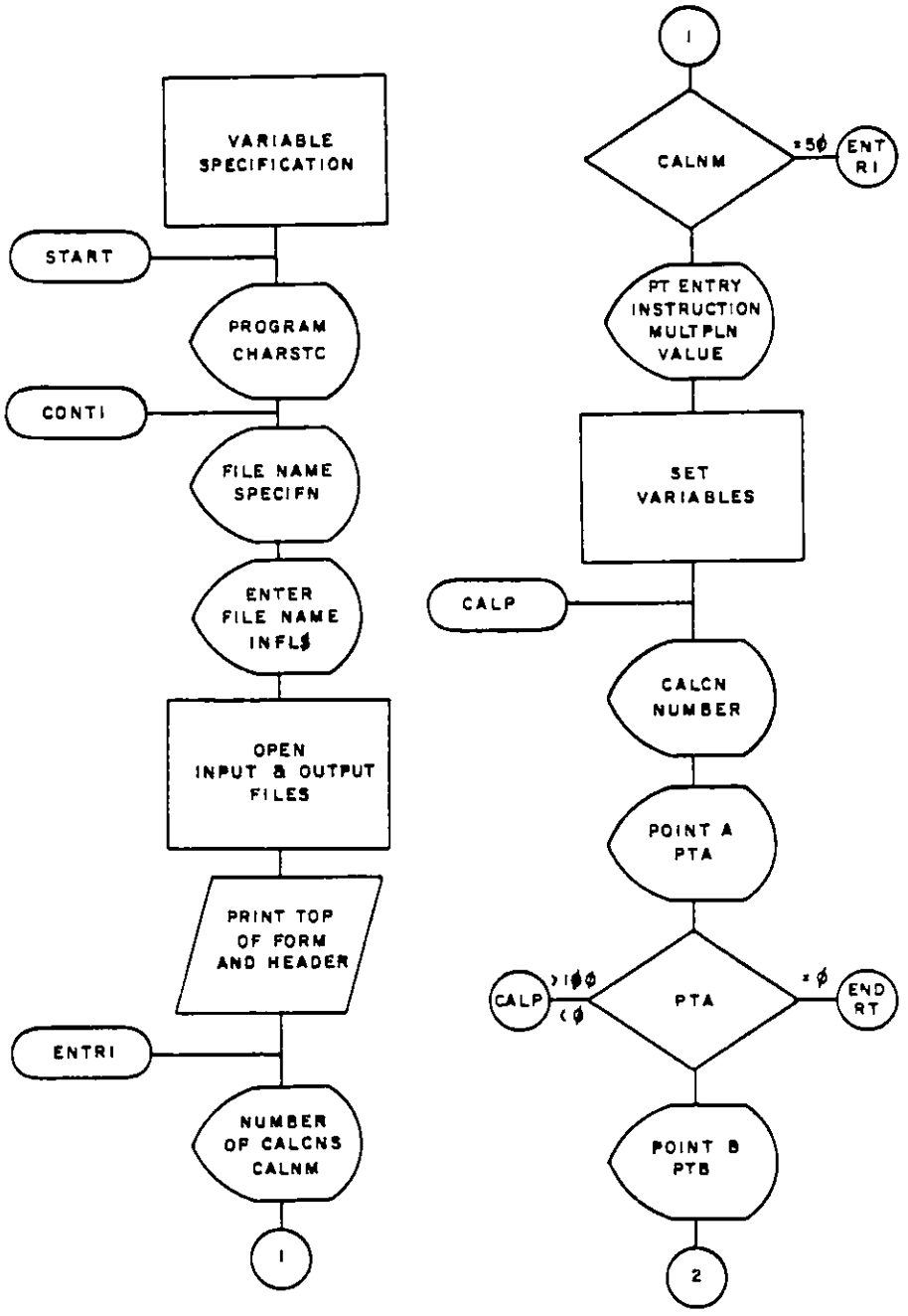
NXRCD : LET SUB1 = 1
FNTRD : INPUT #5,CFET,PNB
XYRD :  INPUT #5,XSTR(SUB1),YSTR(SUB1)
        LET SUB1 = SUB1 + 1
        IF SUB1 < 101 GOTO  XYRD
        LET SUB1 = 1
FILRD : INPUT #5,FILL
        LET SUB1 = SUB1 + 1
        IF SUB1 < 21 GOTO FILRD
        IF EOF(5) = 1 GOTO ENDRT
        LET SUB1 = 1
CAL1 :  LET SUBA = A(SUB1)
        LET SUBB = B(SUB1)
        LET XASV = XSTR(SUBA)
        LET YASV = YSTR(SUBA)
        LET XBSV = XSTR(SUBB)
        LET YBSV = YSTR(SUBB)
        LET DIFSV = SQR((XASV - XBSV)**2 + (YASV - YBSV)**2)
        LET L = DIFSV/200
        LET L = L * 25.4
        LET IN(SUB1) = L * VALX
        IF SUB1 = CALNM GOTO PRTRT
        LET SUB1 = SUB1 +1
        GOTO CAL1

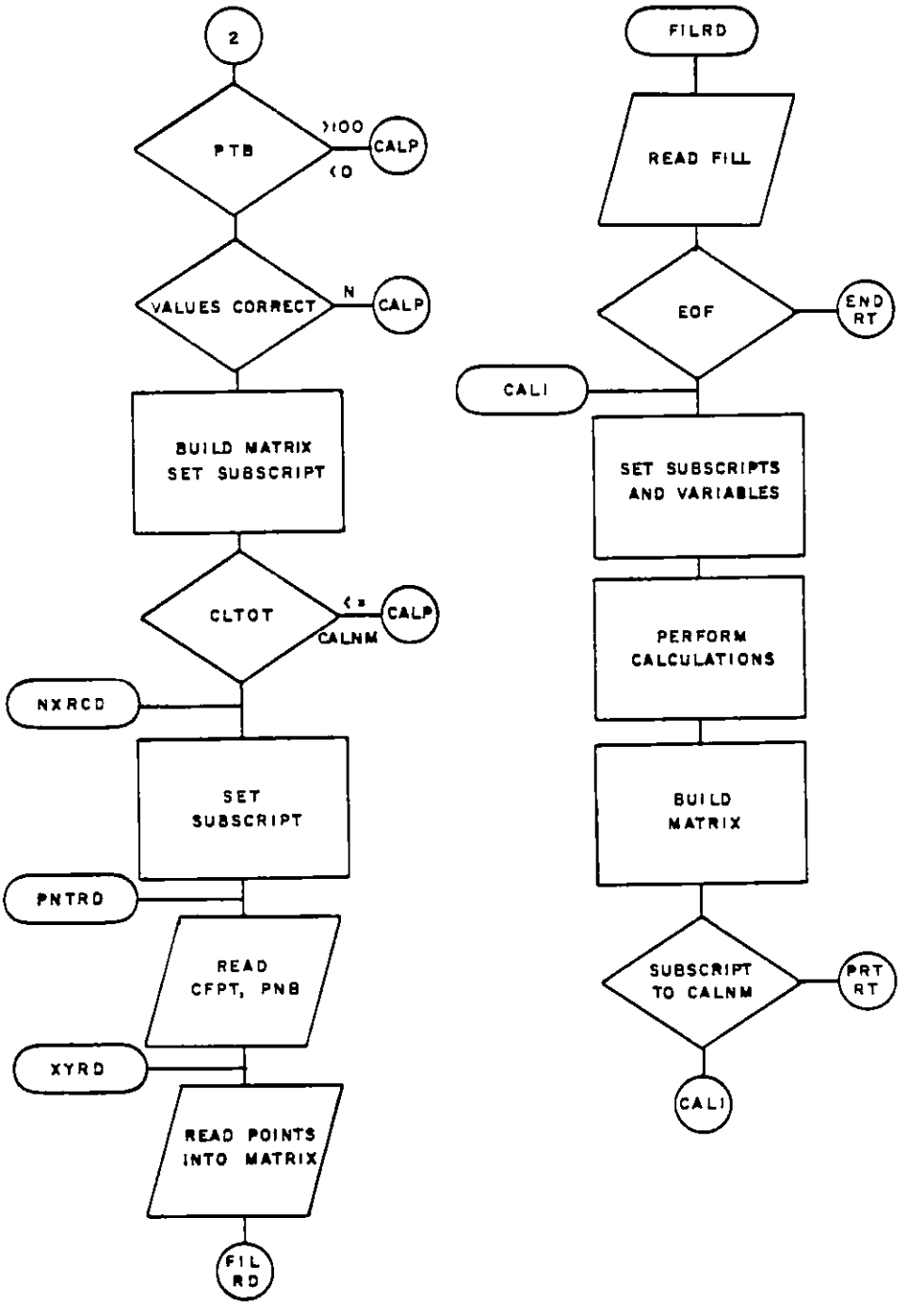

PRTRT : PRINT #7,PNB
        LET SUB1 = 1
PRT2 :  PRINT #7,TAB(10);"POINT";A(SUB1);"TO";B(SUB1);"MULTIPLIED BY";
        PRINT #7,VALX;"EQUALS ";IN(SUB1);" mm."
```

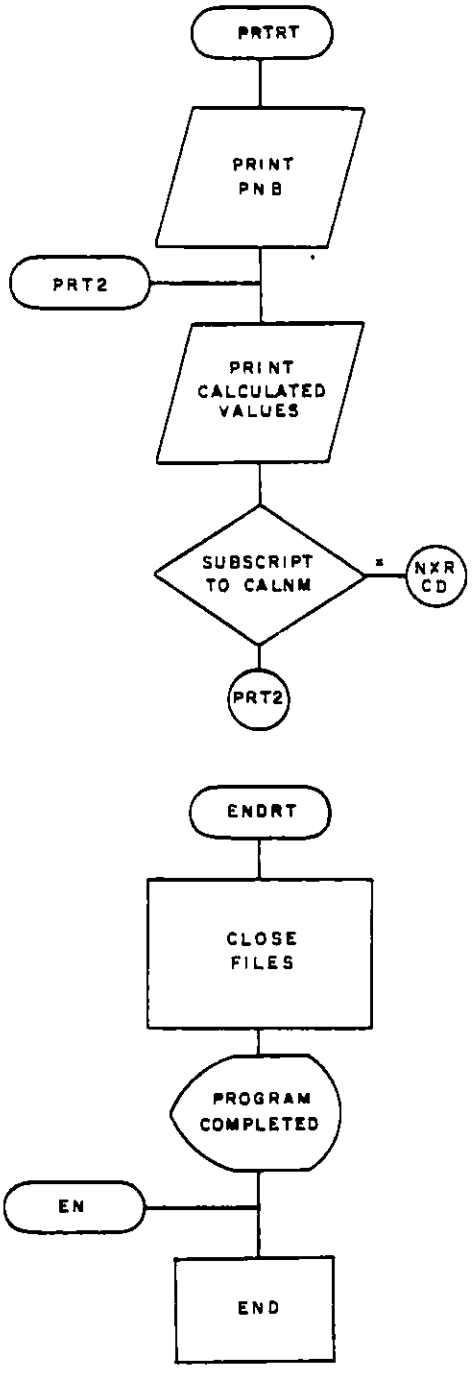
File DEM1:MULDET.BAS[1,4] printed on 0/0/0

```
        IF SUB1 = CALNM GOTO NXRCD
        LET SUB1 = SUB1 + 1
        GOTO PRT2

ENDRT : CLOSE #5
        CLOSE #7
        PRINT TAB(-1,0);TAB(15,5)
        PRINT "THE DISTANCE MULTIPLICATION PROGRAM IS COMPLETE.";TAB(1a,5)
        INPUT "PRESS RETURN FOR NEXT PROGRAM",ANS$
EN :    END
```


VARIABLE SPECIFICATION
START
PROGRAM CHARSTC
CONTI
FILE NAME SPECIFN
ENTER FILE NAME INFL$
OPEN INPUT & OUTPUT FILES
PRINT TOP OF FORM AND HEADER
ENTRI
NUMBER OF CALCNS CALNM
1
1
CALNM
=5Ø
ENT RI
PT ENTRY INSTRUCTION MULTPLN VALUE
SET VARIABLES
CALP
CALCN NUMBER
POINT A PTA
CALP
>1ØØ
<Ø
PTA
=Ø
END RT
POINT B PTB
2


2
PTB
>100
<0
CALP
VALUES CORRECT
N
CALP
BUILD MATRIX
SET SUBSCRIPT
CLTOT
<=
CALNM
CALP
NXRCD
SET
SUBSCRIPT
PNTRD
READ
CFPT, PNB
XYRD
READ POINTS
INTO MATRIX
FIL
RD
FILRD
READ FILL
EOF
END
RT
CALI
SET SUBSCRIPTS
AND VARIABLES
PERFORM
CALCULATIONS
BUILD
MATRIX
SUBSCRIPT
TO CALNM
PRT
RT
CALI


PRTRT
PRINT
PNB
PRT2
PRINT
CALCULATED
VALUES
SUBSCRIPT
TO CALNM
=
NXR
CD
PRT2
ENDRT
CLOSE
FILES
PROGRAM
COMPLETED
EN
END

APPENDIX VII

ADDIST.BAS - Program and Flowchart

File DB1:ADDIST.BAS[1,4] printed on 0/0/0

```
'         ************************************************************
'         *                                                          *
'         *    ADDIST.BAS - THIS PROGRAM WILL CALCULATE THE          *
'         *    DISTANCE BETWEEN POINTS AND ADD THE VALUES FOR A      *
'         *    MINIMUM OF THREE POINTS AND A MAXIMUM OF FIVE.        *
'         *                                                          *
'         ************************************************************
'
'
          STRSIZ 20
          MAP1 PNBR
             MAP2 PNB,F,6
          MAP1 XYSTR(100)
             MAP2 XSTR,B,2
             MAP2 YSTR,B,2
          MAP1 AB(50)
             MAP2 A,F
             MAP2 B,F
             MAP2 C,F
             MAP2 D,F
             MAP2 E,F
          MAP1 LIN(50)
             MAP2 IN,F,6
          MAP1 CEPT
             MAP2 CEPT,F,6
          MAP1 FILR
             MAP2 FILL,3,5
          MAP1 CALPTR(50)
             MAP2 CALPT,F
'
'
          FLG = 0
'
'
START :   PRINT TAB(-1,0)
          PRINT "THIS PROGRAM WILL CALCULATE DISTANCES BETWEEN POINTS"
          PRINT "AND WILL THEN ADD THE VALUES TOGETHER TO GIVE A"
          PRINT "TOTAL VALUE. A MINIMUM OF THREE POINTS IS REQUIRED"
          PRINT "AND A MAXIMUM OF FIVE POINTS MAY BE USED. A MAXIMUM"
          PRINT "OF (50) CALCULATIONS MAY BE EXECUTED PER RECORD."
          PRINT " "
          INPUT "DO YOU WISH TO USE THIS PROGRAM (Y OR N) :",AN1$
          IF AN1$ = "N" GOTO EN
          IF AN1$ = "Y" GOTO CONT
          GOTO START
'
CONT :    PRINT TAB(-1,0);TAB(5,5)
          PRINT "ENTER THE NAME OF THE DATA FILE THAT IS TO BE USED.";TAB(-,5)
          PRINT "THE FILE NAME MUST BE PRECEDED BY DB1:,";TAB(7,5)
          PRINT "THE NAME MUST BE IMMEDIATELY FOLLOWED WITH";TAB(-,5)
          PRINT "A PERIOD AND THE THREE LETTERS DAT. FOR EXAMPLE";TAB(9,5)
          PRINT "IF THE FILE NAME IS CEPH ONE SHOULD ENTER";TAB(10,5)
          PRINT "DB1:CEPH.DAT FOR THE FILE NAME. YOUR FILE";TAB(11,5)
          PRINT "NAME WILL BE THE SAME AS CREATED BY THE";TAB(12,5)
          PRINT "PLTLOD PROGRAM.";TAB(14,5)
          INPUT "ENTER THE FILE NAME : ",INFL$

          OPEN #5,INFL$,INPUT
          OPEN #7,"TRM:PRINT",OUTPUT
          PRINT #7,CHR(12)
```

```
File DB1:ADDIST.BAS[1,4] printed on 0/0/0

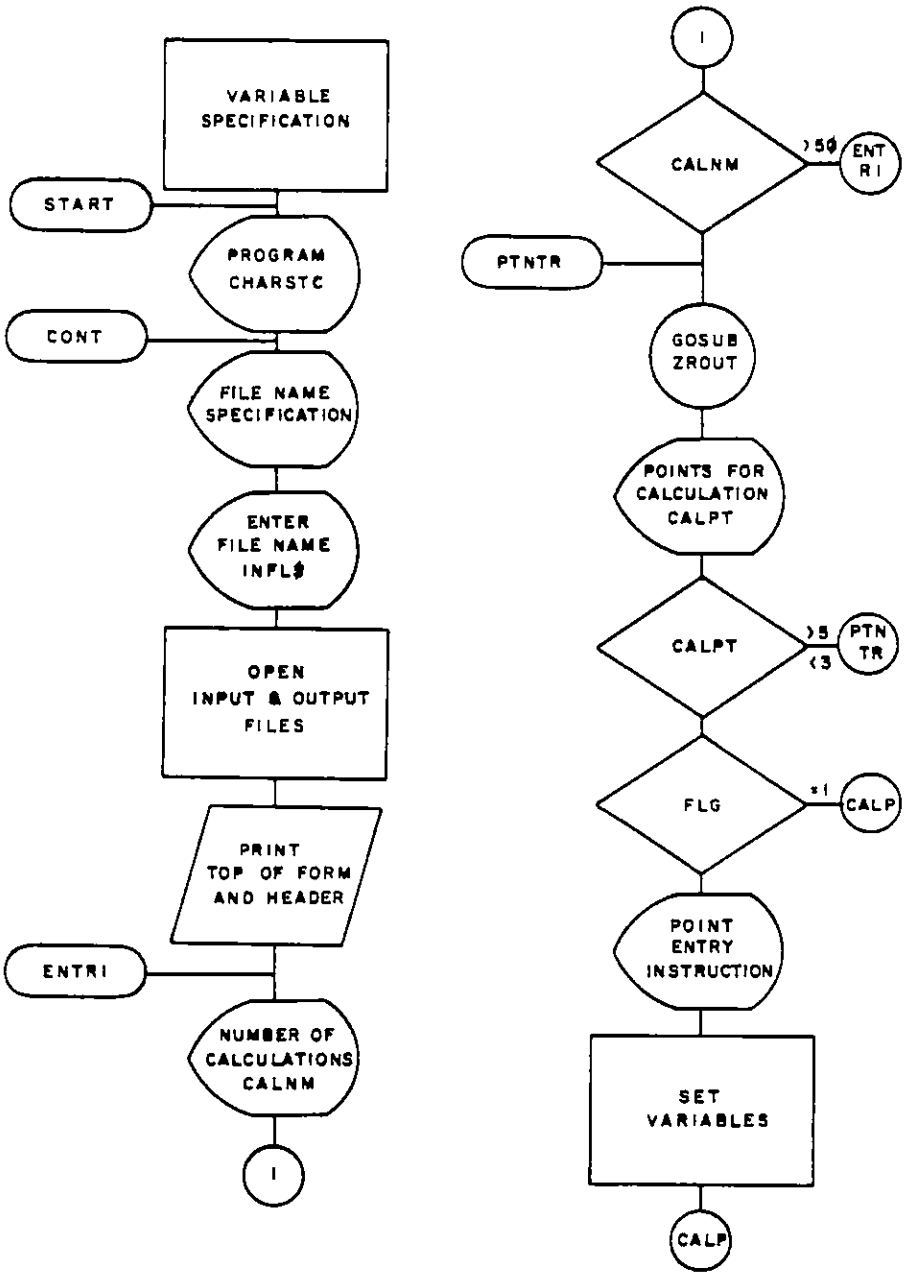
          PRINT #7,TAB(40);"ADDITION OF DISTANCE CALCULATIONS"
          SUBI = 1
'
ENTR1 :   PRINT TAB(-1,0);TAB(15,5)
          INPUT "ENTER THE NUMBER OF CALCULATIONS (MAX = 50) :",CALNM
          IF CALNM > 50 GOTO ENTR1
PTNTR :   GOSUB ZROUT
          PRINT TAB(-1,0);TAB(15,5)
          INPUT "ENTER THE NUMBER OF POINTS FOR CALCULATION :",CALPT(SUBI)
          IF CALPT(SUBI) > 5 GOTO PTNTR'
          IF CALPT(SUBI) < 3 GOTO PTNTR'
          IF FLG = 1 GOTO CALP
          PRINT TAB(-1,0)
          PRINT "THE POINTS WILL BE REQUESTED IN TERMS OF THE LETTERS"
          PRINT "(A) THROUGH (E), DEPENDING UPON THE NUMBER OF POINTS"
          PRINT "DESIRED PER CALCULATION. THE SEQUENCE NUMBER OF EACH"
          PRINT "SHOULD BE THE RESPONSE. IF IT IS DESIRED TO EXIT"
          PRINT "FROM THE PROGRAM ENTER A(0) FOR POINT (A).'
          PRINT " "
          INPUT "PRESS RETURN TO CONTINUE.",ANS$
          SUB1 = 1
          CLTOT = 1
          FLG = 1
CALP :    PRINT TAB(-1,0);TAB(15,5)
          PRINT "THIS IS CALCULATION NUMBER :";CLTOT;TAB(17,5)
          INPUT "ENTER POINT (A) : ",PTA
          IF PTA > 100 GOTO CALP
          IF PTA = 0 GOTO ENDPT'
          IF PTA < 0 GOTO CALP
          PRINT " ";TAB(13,5)
          INPUT "ENTER POINT (B) : ",PTB
          IF PTB > 100 GOTO CALP
          IF PTB < 1 GOTO CALP'
          PRINT " ";TAB(17,5)
          INPUT "ENTER POINT (C) : ",PTC
          IF PTC > 100 GOTO CALP
          IF PTC < 1 GOTO CALP
          IF CALPT(SUBI) < 4 GOTO ETMRX
          PRINT " ";TAB(20,5)
          INPUT "ENTER POINT (D) : ",PTD
          IF PTD > 100 GOTO CALP
          IF PTD < 1 GOTO CALP
          IF CALPT(SUBI) < 5 GOTO ETMRX
          PRINT " ";TAB(21,5)
          INPUT "ENTER POINT (E) : ",PTE
          IF PTE > 100 GOTO CALP
          IF PTE < 1 GOTO CALP
ETMRX :   PRINT " ";TAB(23,5)
          INPUT "ARE ALL VALUES CORRECT : ",COR$
          IF COR$ = "N" GOTO CALP
          A(SUB1) = PTA
          B(SUB1) = PTB : C(SUB1) = PTC
          D(SUB1) = PTD : E(SUB1) = PTE
          CLTOT = CLTOT + 1
          SUB1 = SUB1 + 1
          SUBI = SUBI + 1
          IF CLTOT >= CALNM GOTO PTNTR
'
'
NXPTD :   SUB1 = 1
```

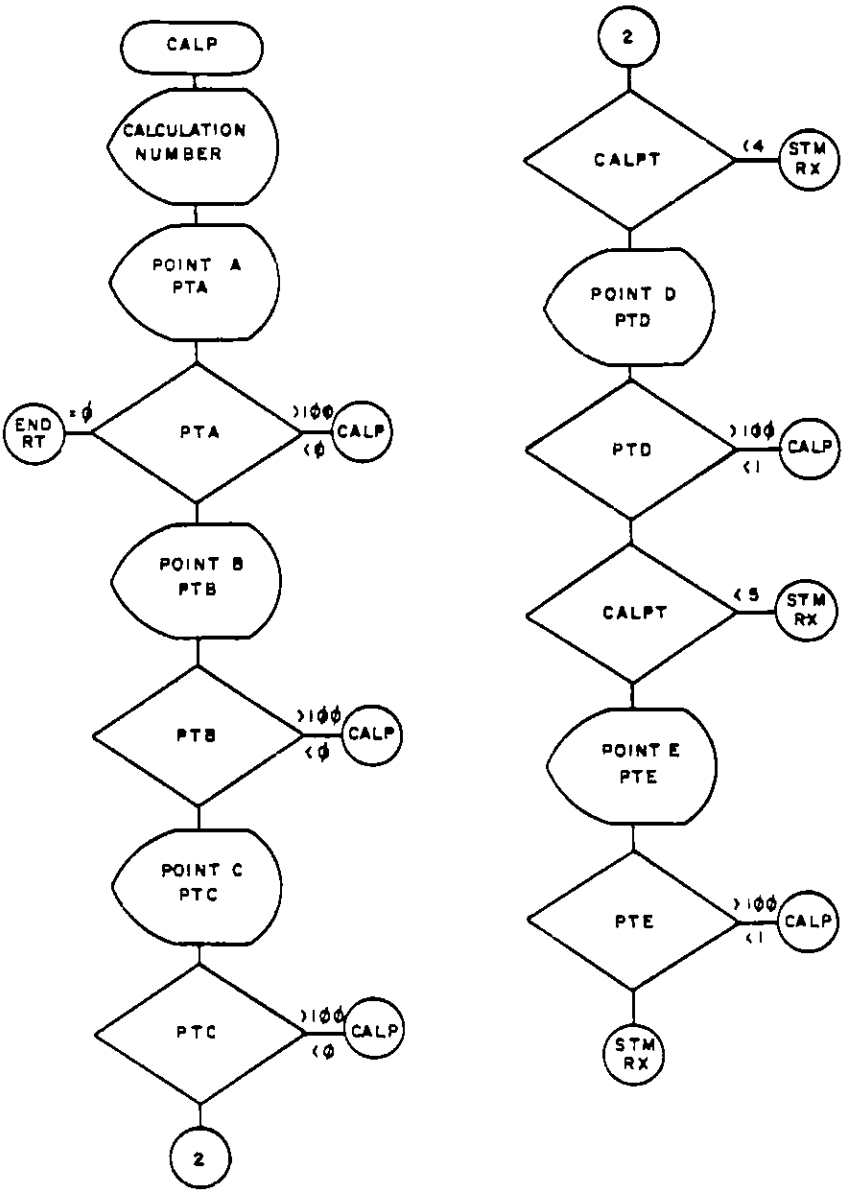
```
File DB1:ADDIST.BAS[1,4] printed on 0/0/0

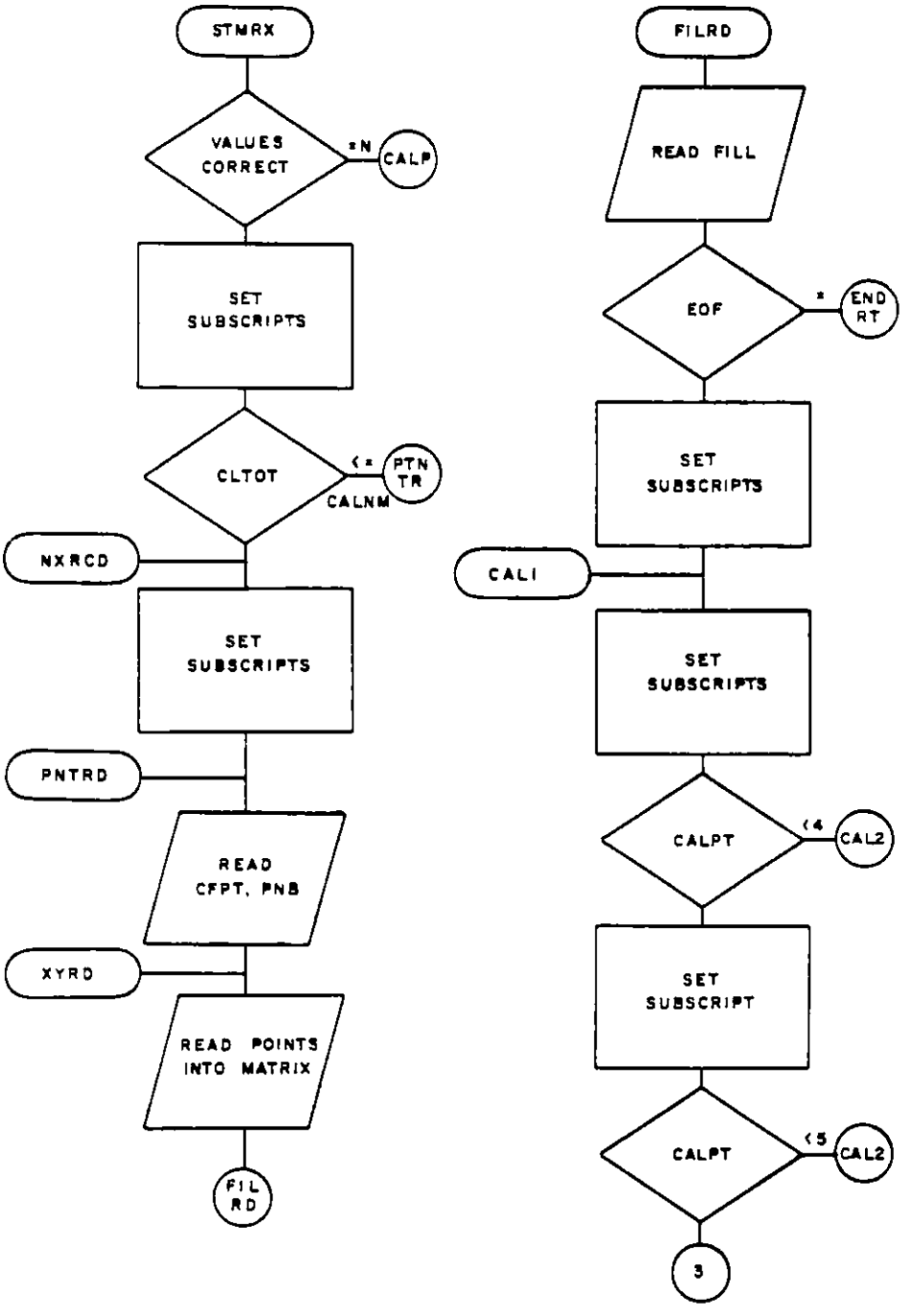
ENTRD :   INPUT #5,CNT,ENP
XYRD :    INPUT #5,XSTR(SUB1),YSTR(SUB1)
          SUB1 = SUB1 + 1
          IF SUB1 < 101 GOTO XYRD
          SUB1 = 1
FILRD :   INPUT #5,FILL
          SUB1 = SUB1 + 1
          IF SUB1 < 21 GOTO FILRD
          IF EOF(5) = 1 GOTO ENDRT
          SUB1 = 1
          SUB2 = 1
!
!
CAL1 :    SUBA = A(SUB1)
          SUBB = B(SUB1)
          SUBC = C(SUB1)
          IF CALPT(SUB2) < 4 GOTO CAL2
          SUBD = D(SUB1)
          IF CALPT(SUB2) < 5 GOTO CAL2
          SUBE = E(SUB1)
CAL2 :    XASV = XSTR(SUBA)
          YASV = YSTR(SUBA)
          XBSV = XSTR(SUBB)
          YBSV = YSTR(SUBB)
          XCSV = XSTR(SUBC)
          YCSV = YSTR(SUBC)
          IF CALPT(SUB2) < 4 GOTO CAL3
          XDSV = XSTR(SUBD)
          YDSV = YSTR(SUBD)
          IF CALPT(SUB2) < 5 GOTO CAL3
          XESV = XSTR(SUBE)
          YESV = YSTR(SUBE)
CAL3 :    DIFSV1 = SQR((XASV-XBSV)**2+(YASV-YBSV)**2)
          DIFSV2 = SQR((XBSV-XCSV)**2+(YBSV-YCSV)**2)
          ADSV = DIFSV1 + DIFSV2
          IF CALPT(SUB2) < 4 GOTO CAL4
          DIFSV3 = SQR((XCSV-XDSV)**2+(YCSV-YDSV)**2)
          ADSV = ADSV + DIFSV3
          IF CALPT(SUB2) < 5 GOTO CAL4
          DIFSV4 = SQR((XDSV-XESV)**2+(YDSV-YESV)**2)
          ADSV = ADSV + DIFSV4
CAL4 :    L = ADSV/2.0
          L = L * 25.4
          IN(SUB1) = L
          IF SUB1 = CALNM GOTO PRTRT
          SUB1 = SUB1 + 1
          SUB2 = SUB2 + 1
          GOTO CAL1
!
!
PRTRT :   PRINT #7,ENP
          SUB1 = 1
PRT2 :    PRINT #7,TAB(10);"TOTAL DISTANCE OF POINTS :";A(SUB1);",";
          PRINT #7,B(SUB1);",";C(SUB1);",";D(SUB1);",";E(SUB1);" CALC= ";
          PRINT #7,IN(SUB1);" mm."
          IF SUB1 = CALNM GOTO NXRCD
          SUB1 = SUB1 + 1
          GOTO PRT2
!
!
```

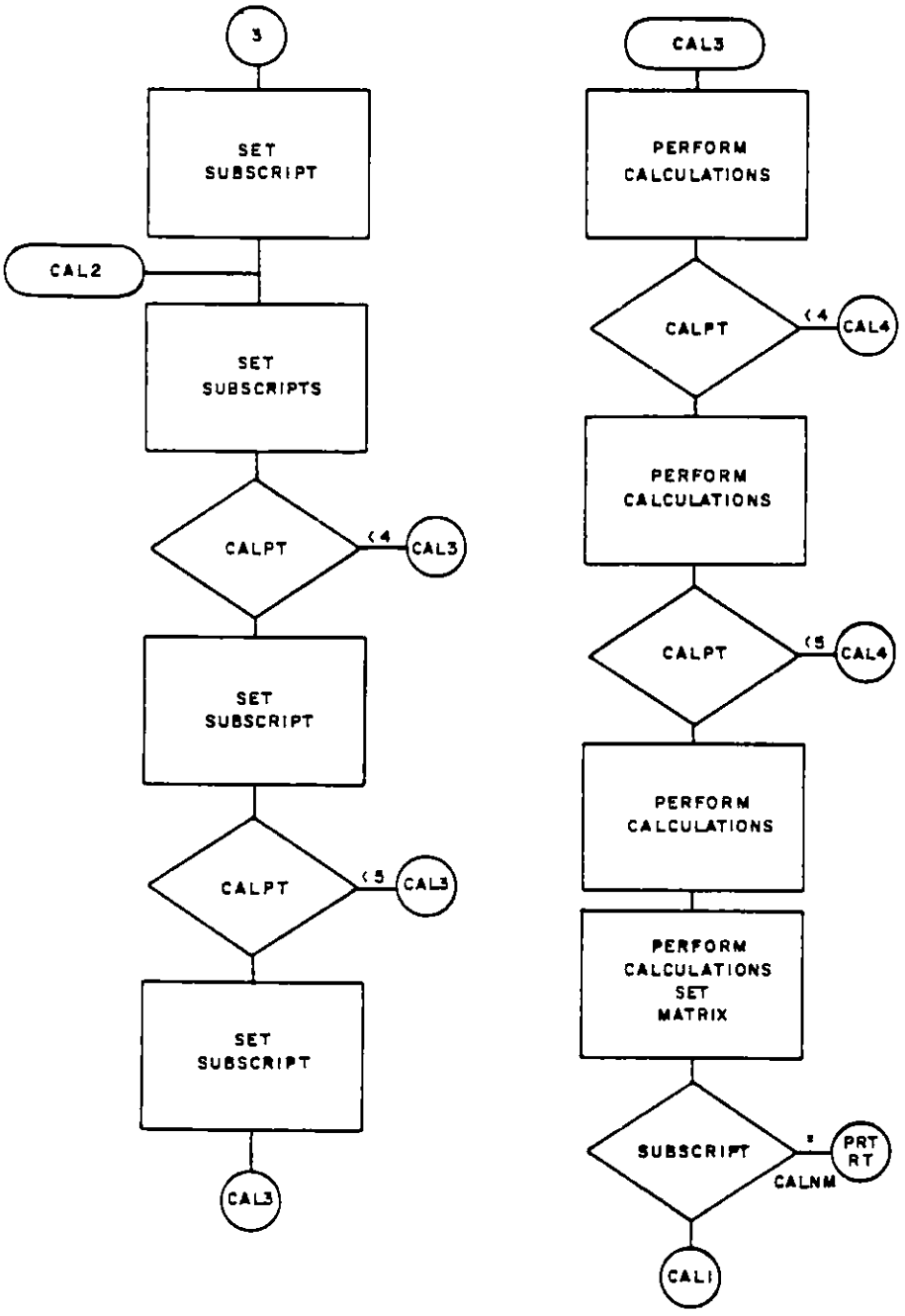
File DSK1:ADDIST.BAS[1,4] printed on 0/0/0

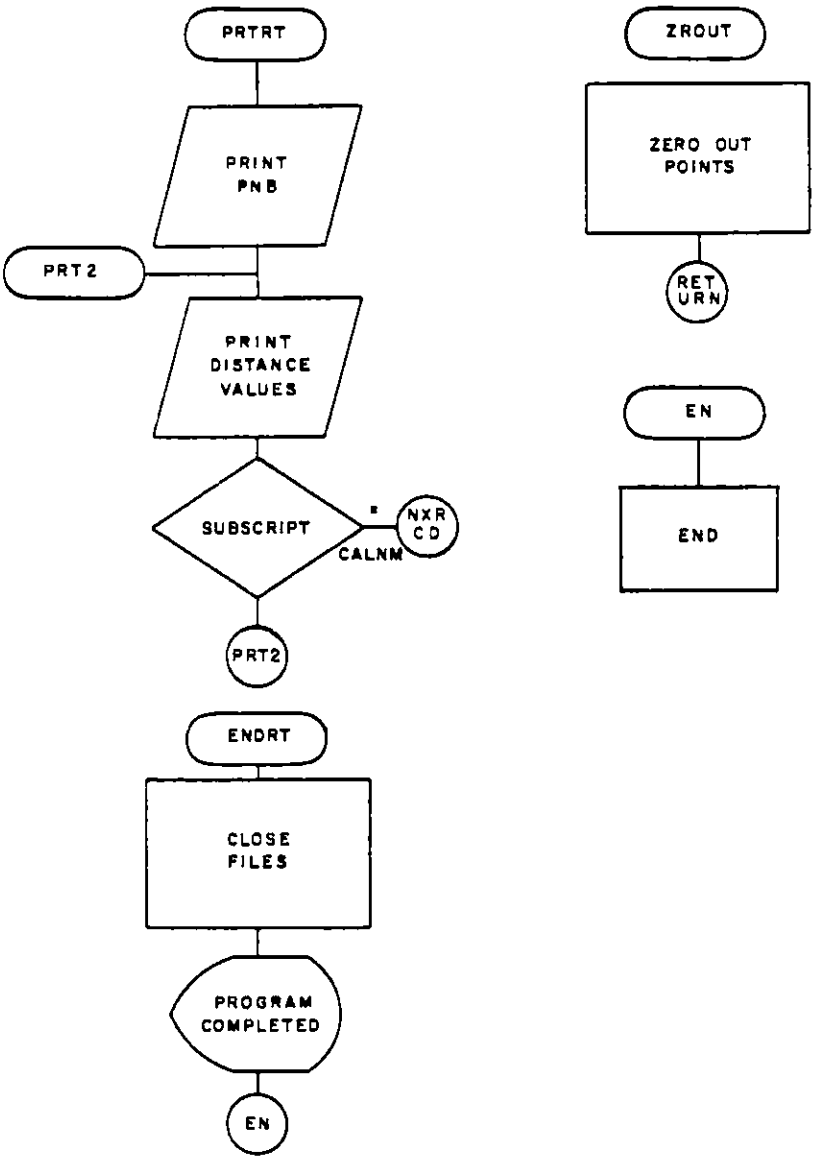
```
ENDRT :   CLOSE #5
          CLOSE #7
          PRINT TAB(-1,0);TAB(15,5)
          PRINT "THE ADD DISTANCE PROGRAM IS COMPLETE.";TAB(16,5)
          INPUT "PRESS RETURN FOR NEXT PROGRAM",ANS$
          GOTO EN
ZROUT :   PTA = 0 : PTB = 0 : PTC = 0 : PTD = 0 : PTE = 0
          RETURN
EN :      END
```


VARIABLE SPECIFICATION
START
PROGRAM CHARSTC
CONT
FILE NAME SPECIFICATION
ENTER FILE NAME INFL$
OPEN INPUT & OUTPUT FILES
PRINT TOP OF FORM AND HEADER
ENTRI
NUMBER OF CALCULATIONS CALNM
I
I
CALNM
>50
ENTRI
PTNTR
GOSUB ZROUT
POINTS FOR CALCULATION CALPT
CALPT
>5
<3
PTNTR
FLG
=1
CALP
POINT ENTRY INSTRUCTION
SET VARIABLES
CALP


CALP
CALCULATION NUMBER
POINT A PTA
PTA
END RT
=ϕ
>100
<ϕ
CALP
POINT B PTB
PTB
>100
<ϕ
CALP
POINT C PTC
PTC
>100
<ϕ
CALP
2
2
CALPT
<4
STM RX
POINT D PTD
PTD
>100
<1
CALP
CALPT
<5
STM RX
POINT E PTE
PTE
>100
<1
CALP
STM RX


STMRX
VALUES CORRECT
=N
CALP
SET SUBSCRIPTS
CLTOT
<=
PTN TR
CALNM
NXRCD
SET SUBSCRIPTS
PNTRD
READ CFPT, PNB
XYRD
READ POINTS INTO MATRIX
FIL RD
FILRD
READ FILL
EOF
*
END RT
SET SUBSCRIPTS
CALI
SET SUBSCRIPTS
CALPT
<4
CAL2
SET SUBSCRIPT
CALPT
<5
CAL2
3


3
SET SUBSCRIPT
CAL2
SET SUBSCRIPTS
CALPT
<4
CAL3
SET SUBSCRIPT
CALPT
<5
CAL3
SET SUBSCRIPT
CAL3
CAL3
PERFORM CALCULATIONS
CALPT
<4
CAL4
PERFORM CALCULATIONS
CALPT
<5
CAL4
PERFORM CALCULATIONS
PERFORM CALCULATIONS SET MATRIX
SUBSCRIPT
=
CALNM
PRT RT
CAL1


PRTRT
PRINT
PNB
PRT 2
PRINT
DISTANCE
VALUES
SUBSCRIPT
*
NXR
CD
CALNM
PRT2
ENDRT
CLOSE
FILES
PROGRAM
COMPLETED
EN
ZROUT
ZERO OUT
POINTS
RET
URN
EN
END

APPENDIX VIII

CANLOD.BAS - Program and Flowchart

File DEV1:CANLOD.BAS[1,4] printed on 07/07/0

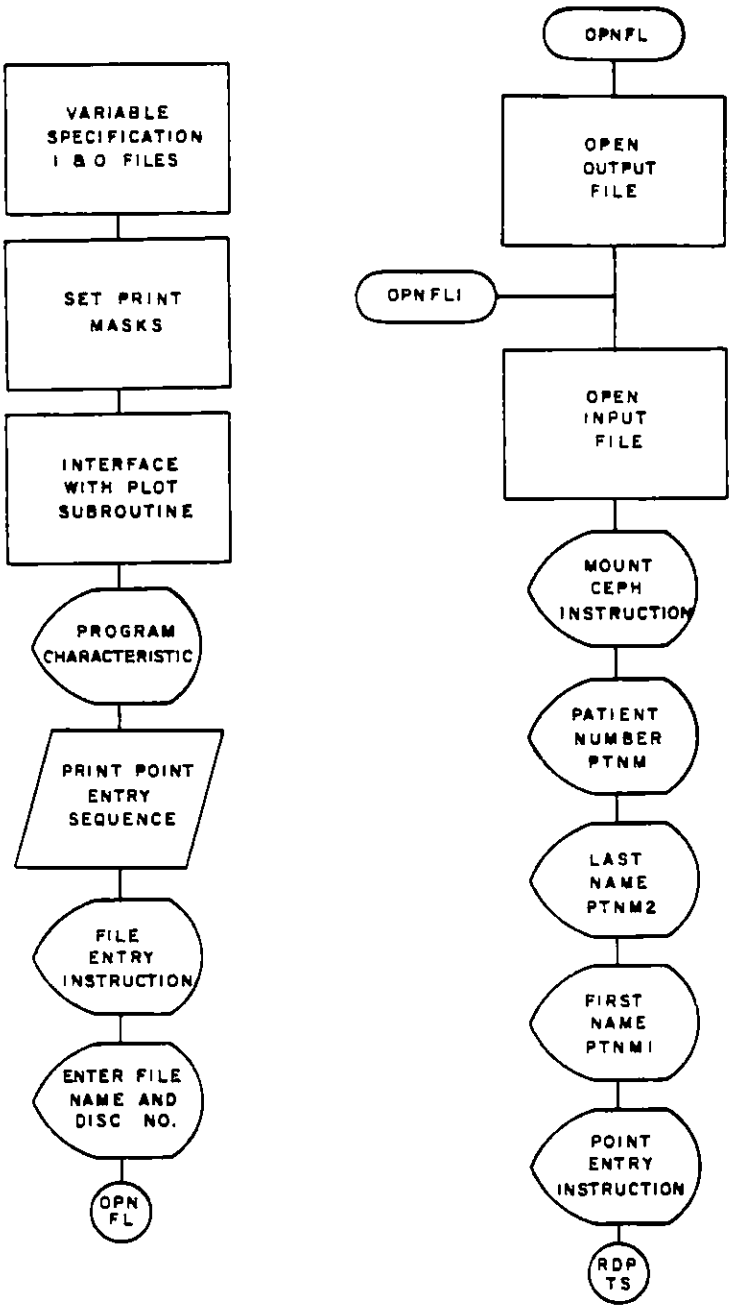
```
'              ************************************************
'              *                                              *
'              *  CANLOD.BAS- THIS PROGRAM TRANSFERS COORDINATE *
'              *  FROM THE DIGITIZER TO A DISC FILE. THE OUTPUT *
'              *  FROM THIS PROGRAM IS THE INPUT FOR THE CANNAL *
'              *  PROGRAM.                                    *
'              *                                              *
'              ************************************************
'
'
            STRSIZ 32
            MAP1 PNBR
               MAP2 PNB,F,6
            MAP1 PNAM
               MAP2 PTNM1,S,10
               MAP2 PTNM2,S,16
            MAP1 XYOUT(16)
               MAP2 XOUT,B,2
               MAP2 YOUT,B,2
            MAP1 XY
               MAP2 X,B,2
               MAP2 Y,B,2
               MAP2 FLAG,B,2
'
            TVAR$ = "TTTT"
            ESC$ ="\                \"
            ESI$ ="\         \"
            CEFN = 0
            XCALL FLOT,"TRMB"
            PRINT TAB(-1,0);TAB(5,5)
            PRINT "THIS PROGRAM WILL TRANSFER CEPHALOMETRIC POINTS";TAB(6,5)
            PRINT "FROM THE DIGITIZER TO A DISC FILE. THE OPERATOR";TAB(7,5)
            PRINT "WILL ENTER THE NAME OF THE FILE TO BE GENERATED";TAB(8,5)
            PRINT "UPON REQUEST. THE OPERATOR WILL THEN ENTER THE ";TAB(9,5)
            PRINT "PATIENT NUMBER AND PATIENT NAME FOR EACH CEPH.";TAB(10,5)
            PRINT "WHEN THE CEPH IS MOUNTED, SIXTEEN POINTS WILL ";TAB(11,5)
            PRINT "BE ENTERED IN THE ORDER SPECIFIED ON THE PRINT-OUT."
            PRINT " ";TAB(13,5)
            INPUT "PLEASE RETURN TO CONTINUE.",AN$
            OPEN #7,"TRM:PRINT",OUTPUT
            PRINT #7,"1. SELLA ; 2. NASION ; 3. ORBITALE ; 4. A-POINT"
            PRINT #7,"5. B-POINT ; 6. PROGNION ; 7. PNATHION ; 8. MENTON"
            PRINT #7,"9. APICAL U1 ; 10. INCISAL U1 ; 11. INCISAL L1 ; "
            PRINT #7,"12. APICAL L1"
            PRINT #7,"13. INCISOR DIFFERANCE POINT ; 14. MOLAR POINT ; "
            PRINT #7,"15. GONION TANGENCY ; 16. PORION"
            CLOSE #7
            PRINT TAB(-1,0);TAB(5,5)
            PRINT "ENTER THE NAME OF THE DATA FILE THAT IS TO BE";TAB(6,5)
            PRINT "CREATED. THE NAME CAN BE ANY ALPHA-NUMERIC ";TAB(7,5)
            PRINT "COMBINATION, BUT MUST START WITH A LETTER.";TAB(8,5)
            PRINT "THE FILE NAME MUST BE PRECEDED BY DEV1:.";TAB(9,5)
            PRINT "THE NAME MUST BE IMMEDIATELY FOLLOWED WITH";TAB(10,5)
            PRINT "A PERIOD AND THE THREE LETTERS DAT. FOR ";TAB(11,5)
            PRINT "EXAMPLE TO NAME A FILE CEPH ONE SHOULD";TAB(12,5)
            PRINT "ENTER DEV1:CEPH.DAT FOR THE FILE NAME.";TAB(13,5)
            INPUT "ENTER THE FILE NAME : ",INFL$
            PRINT " ";TAB(15,5)
            INPUT "ENTER THE DISC NUMBER : ",DIENM
OPENFL :    OPEN #5,INFL$,OUTPUT
```

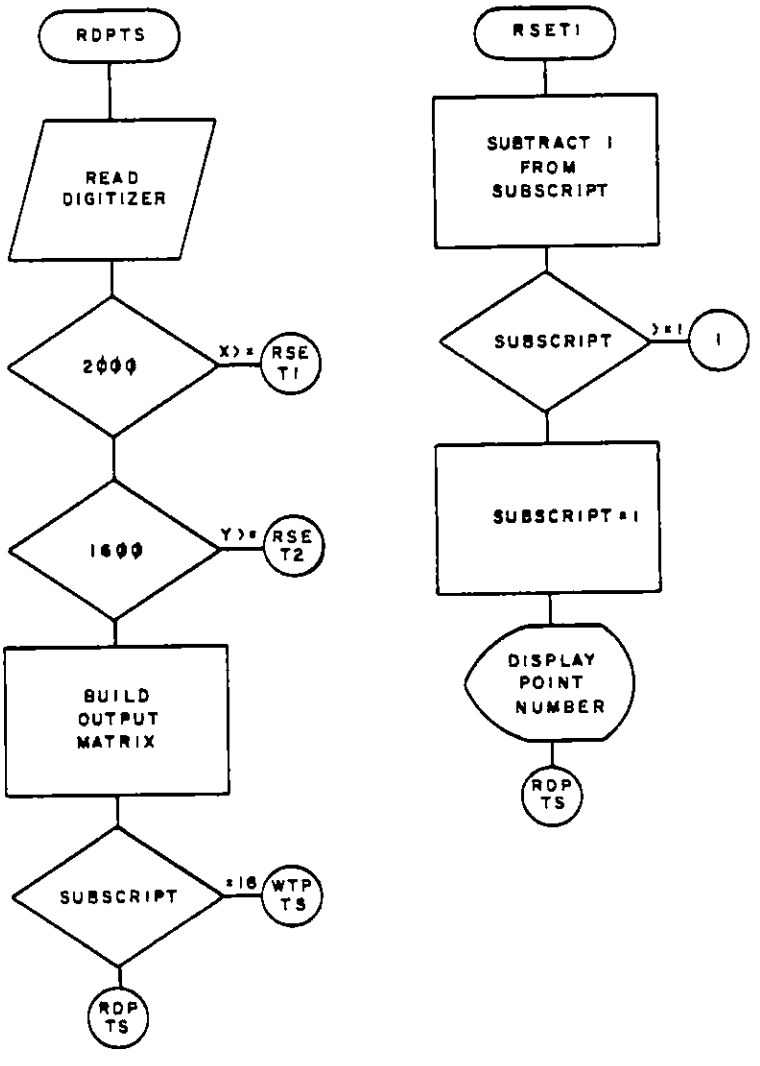
```
File DB1:CANLOD.BAS[1,4] Printed on 0/0/0

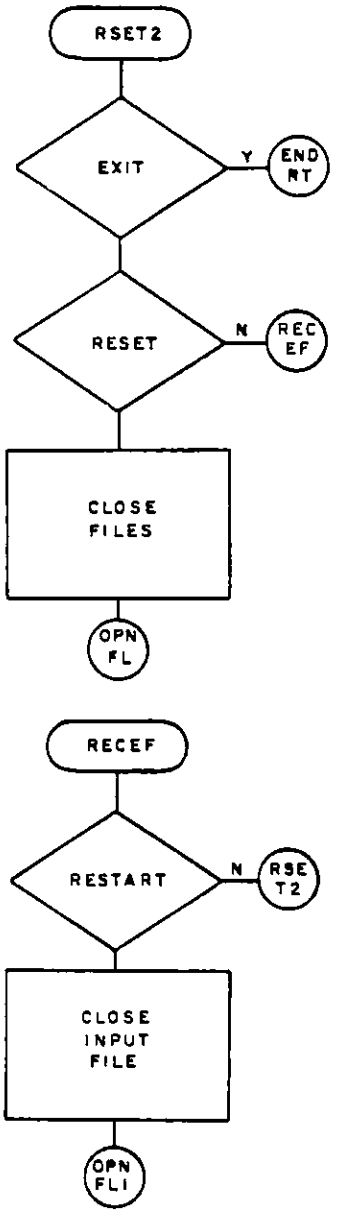
OPNFL1 :  OPEN #3,"TRM:BITRAD",INPUT
          PRINT TAB(-1,0);TAB(5,5)
          PRINT "MOUNT CEPH TRACING. LINE UP THE EDGES WITH";TAB(4,5)
          PRINT "THE MARKS ON THE DIGITIZER."
          PRINT " ";TAB(7,5)
          PRINT "TO EXIT FROM THE PROGRAM ENTER 0 FOR PATIENT NUMBER."
          PRINT " ";TAB(11,5)
          INPUT "ENTER A 6-DIGIT PATIENT NUMBER : ",PNB
          IF PNB = 0 GOTO ENDRT
          PRINT " ";TAB(13,5)
          INPUT "ENTER LAST NAME (MAX = 16 LETTERS) : ",PTNM2
          PRINT " ";TAB(15,5)
          INPUT "ENTER FIRST NAME (MAX = 10 LETTERS) : ",PTNM1
          PRINT TAB(-1,0);TAB(5,5)
          PRINT "POINTS MAY ONLY BE ENTERED FROM WITHIN";TAB(6,5)
          PRINT "THE BOUNDARIES OF THE MARKS ON THE DIGITIZER.";TAB(7,5)
          PRINT "IF A MISTAKE IS MADE, PRESS THE STYLUS IN THE";TAB(8,5)
          PRINT "RIGHT-HAND BOUNDARY AREA. THIS WILL CAUSE THE";TAB(9,5)
          PRINT "PREVIOUS ENTRY TO BE ERASED. TO RESET THE ";TAB(10,5)
          PRINT "PROGRAM, RESTART A TRACING,OR EXIT FROM THE";TAB(11,5)
          PRINT "PROGRAM PRESS THE STYLUS ABOVE THE UPPER";TAB(12,5)
          PRINT "BOUNDARY. POINTS MAY NOW BE ENTERED."
          SUB1 = 1
RDPTS :   INPUT #3,X,Y,FLAG
          IF X <= 2000 GOTO RSET1
          IF Y >= 1600 GOTO RSET2
          XOUT(SUB1) = X
          YOUT(SUB1) = Y
          IF SUB1 = 16 GOTO WTPTS
          SUB1 = SUB1 + 1
          GOTO RDPTS
RSET1 :   SUB1 = SUB1 - 1
          IF SUB1 < 1 SUB1 = 1
          PRINT TAB(13,5);TAB(-1,0);TAB(15,5);
          PRINT "YOU ARE NOW ENTERING POINT # :";SUB1
          GOTO RDPTS
RSET2 :   SUB1 = 1
          PRINT TAB(-1,0);TAB(10,5)
          INPUT "DO YOU WISH TO EXIT FROM THE PROGRAM (Y OR N) :",ANA$
          IF ANA$ = "Y" GOTO ENDRT
          PRINT TAB(-1,0);TAB(10,5)
          INPUT "DO YOU WISH TO RESET THE PROGRAM (Y OR N) :",ANB$
          IF ANB$ = "N" GOTO RECEF
          CLOSE #3
          CLOSE #5
          GOTO OPNFL
RECEF :   PRINT TAB(-1,0);TAB(10,5)
          INPUT "DO YOU WISH TO RESTART THE TRACING (Y OR N) :",ANC$
          IF ANC$ = "N" GOTO RSET2
          CLOSE #3
          GOTO OPNFL1
WTPTS :   PRINT #5,PNB
          PRINT #5,USING 51$,PTNM1
          PRINT #5,USING 52$,PTNM2
          FOR SUB1 = 1 TO 16
          PRINT #5,XOUT(SUB1);YOUT(SUB1)
          NEXT SUB1
          FOR SUBT = 1 TO 5
          PRINT #5,TVAR$
          NEXT SUBT
```

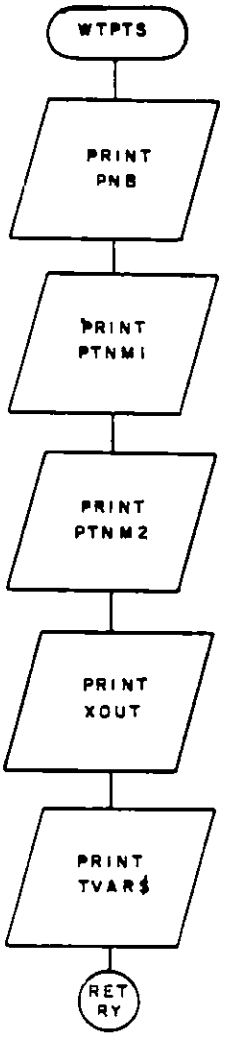
File DSK1:CANLOD.BAS[1,4] Printed on 0/0/0

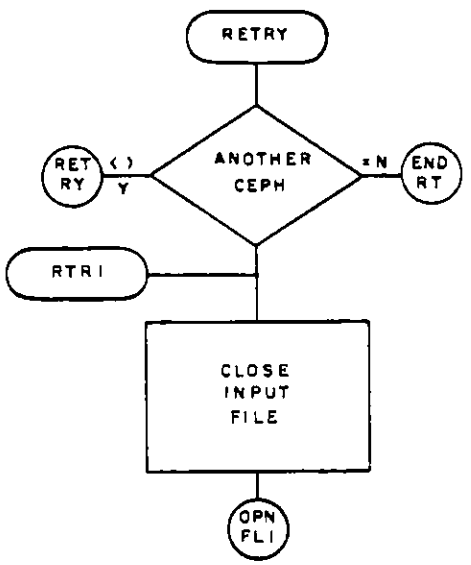
```
            CEFN = CEFN + 1
RETRY :     PRINT TAB(-1,0);TAB(10,5)
            INPUT "DO YOU WISH TO ENTER ANOTHER CEPH :",ANE$
            IF ANE$ = "N" GOTO ENDRT
            IF ANE$ <> "Y" GOTO RETRY
STR1 :      CLOSE #3
            GOTO OPNFL1
ENDRT :     CLOSE #3
            CLOSE #5
            OPEN #7,"TRM:PRINT",OUTPUT
            PRINT #7,CHR(12)
            PRINT #7,TAB(40);"THE NAME OF YOUR FILE IS :";INFL$
            PRINT #7,TAB(40);"YOUR DISC NUMBER IS :";DISNM
            PRINT #7,TAB(40);"YOU HAVE TRANSFERED ";CEFN;"CEPHS TO DISC"
            CLOSE #7
            END
```

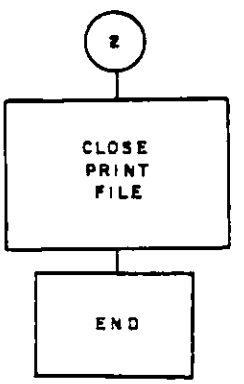

VARIABLE SPECIFICATION I & O FILES
SET PRINT MASKS
INTERFACE WITH PLOT SUBROUTINE
PROGRAM CHARACTERISTIC
PRINT POINT ENTRY SEQUENCE
FILE ENTRY INSTRUCTION
ENTER FILE NAME AND DISC NO.
OPN FL
OPNFL
OPEN OUTPUT FILE
OPNFLI
OPEN INPUT FILE
MOUNT CEPH INSTRUCTION
PATIENT NUMBER PTNM
LAST NAME PTNM2
FIRST NAME PTNMI
POINT ENTRY INSTRUCTION
RDP TS

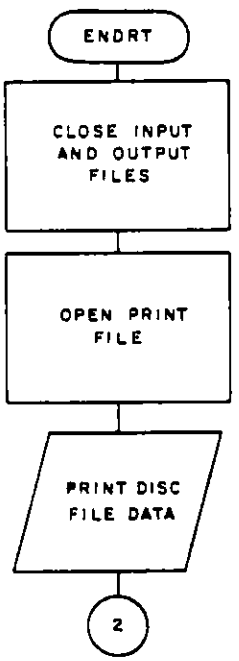

RDPTS
READ DIGITIZER
2000
X>=
RSET1
1600
Y>=
RSET2
BUILD OUTPUT MATRIX
SUBSCRIPT
=16
WTPTS
RDPTS
RSET1
SUBTRACT 1 FROM SUBSCRIPT
SUBSCRIPT
>=1
1
SUBSCRIPT=1
DISPLAY POINT NUMBER
RDPTS


RSET2
EXIT
Y
END RT
RESET
N
REC EF
CLOSE FILES
OPN FL
RECEF
RESTART
N
RSE T2
CLOSE INPUT FILE
OPN FLI


WTPTS
PRINT PNB
PRINT PTNM1
PRINT PTNM2
PRINT XOUT
PRINT TVAR$
RET RY


RETRY
ANOTHER CEPH
RET RY
( )
Y
=N
END RT
RTRI
CLOSE INPUT FILE
OPN FLI


2
CLOSE PRINT FILE
END


ENDRT
CLOSE INPUT AND OUTPUT FILES
OPEN PRINT FILE
PRINT DISC FILE DATA
2

APPENDIX IX

CANNAL.BAS - Program and Flowchart

File DSK1:CANNAL.BAS[1,4] printed on 07/07/

```
!              ************************************************************
!              *                                                          *
!              *   CANNAL.BAS - THIS PROGRAM ENABLES THE OPERATOR         *
!              *   TO PERFORM ANY COMBINATION OF TWEED,DOWNS,             *
!              *   BEGG, AND REIDEL ANALYSES.                             *
!              *                                                          *
!              ************************************************************
!
!
             STRSIZ 32
             MAP1 FNBR
                MAP2 PNB,F,6
             MAP1 XYSTR(16)
                MAP2 XSTR,B,2
                MAP2 YSTR,B,2
             MAP1 PNAM
                MAP2 PTNM1,S,10
                MAP2 PTNM2,S,10
             MAP1 FILL
                MAP2 TVAR,S,4
             MAP1 TWD(3)
                MAP2 RSLTWD,F
             MAP1 DWN(10)
                MAP2 RSLDWN,F
             MAP1 BEG(10)
                MAP2 RSLBEG,F
             MAP1 RDL(5)
                MAP2 RSLRDL,F
!
TWDTA :      DATA 16,3,15,6,8,15,12,11
DWNS :       DATA 3,16,2,6,2,4,4,6
             DATA 2,6,4,5,16,3,15,8
             DATA 16,3,1,7,16,8,14,13
             DATA 10,9,11,12,13,14,11,12
             DATA 8,15,12,11,4,6,10
BEGS :       DATA 1,2,15,3,8,15,12,11
             DATA 1,2,14,13,2,4,4,6
             DATA 3,16,2,6,2,1,8,10
             DATA 1,2,4,1,2,5
             DATA 4,4,11
RDLS :       DATA 16,3,11,12,3,16,2,4
             DATA 1,2,4,1,2,5
!
!
INSTRT :     PRINT TAB(-1,0);TAB(10,5)
             INPUT "PERFORM TWEED ANALYSIS ? ",TWD$
             PRINT " ";TAB(12,5)
             INPUT "PERFORM DOWNS ANALYSIS ? ",DWN$
             PRINT " ";TAB(14,5)
             INPUT "PERFORM BEGG ANALYSIS ? ",BEG$
             PRINT " ";TAB(16,5)
             INPUT "PERFORM REIDEL ANALYSIS ? ",RDL$
             PRINT TAB(-1,0);TAB(10,5)
             INPUT "ENTER YOUR INPUT FILE NAME : ",INFL$
             OPEN #7,"TRM:PRINT",OUTPUT
             OPEN #5,INFL$,INPUT
             PRINT #7,CHR(17)
!
XYRDE :      SUB1 = 1
             INPUT #5,FNB,PTNM1$,PTNM2$
```

```
File DF1:CANNAL.BAS[1,4] printed on 0/0/0

xyRD :      INPUT #5,XSTR(SUB1),YSTR(SUB1)
            SUB1 = SUB1 + 1
            IF SUB1 < 17 GOTO xyRD
            FOR TFL = 1 TO 8
            INPUT #5,TVAR
            NEXT TFL
            IF EOF(5) = 1 GOTO ENDRT
TWEED :     IF TWD$ = "Y" GOTO TWD1
            FOR SUBT = 1 TO 2
            READ A,B,C,D
            NEXT SUBT
            GOTO DOWN
TWD1 :      FOR SUBT = 1 TO 2
            READ A,B,C,D
            SUBA = A : SUBB = B
            SUBC = C : SUBD = D
            GOSUB ANGL4
            RELTWD(SUBT) = DEG
            NEXT SUBT
            RELTWD(3) = 180 - (RELTWD(1) + RELTWD(2))
            GOTO DOWN
DOWN :      IF DWN$ = "Y" GOTO DWN1
            FOR SUBW = 1 TO 9
            READ A,B,C,D
            NEXT SUBW
            READ A,B,C
            GOTO BEGG
DWN1 :      FOR SUBW = 1 TO 9
            READ A,B,C,D
            SUBA = A : SUBB = B
            SUBC = C : SUBD = D
            GOSUB ANGL4
            ON SUBW GOSUB CH1 ,CH4 ,CH5 ,CH1 ,CH1 ,CH8 ,CH1 ,CH10 ,CH11
            RELDWN(SUBW) = DEG
            NEXT SUBW
            READ A,B,C
            SUBA = A : SUBB = B : SUBC = C
            GOSUB RTLIN
            IF LM = 0 GOTO DWN2
            GOSUB CH12
DWN2 :      RELDWN(SUBW) = LM
            GOTO BEGG
BEGG :      IF BEG$ = "Y" GOTO BEG1
            FOR SUBG = 1 TO 5
            READ A,B,C,D
            NEXT SUBG
            READ A
            GOTO REIDEL
BEG1 :      FOR SUBG = 1 TO 6
            READ A,B,C,D
            SUBA = A : SUBB = B
            SUBC = C : SUBD = D
            GOSUB ANGL4
            ON SUBG GOSUB CH1 ,CH1 ,CH1 ,CH4 ,CH1 ,CH1
            RELBEG(SUBG) = DEG
            NEXT SUBG
            FOR SUBG = 7 TO 8
            READ A,B,C
            SUBA = A : SUBB = B : SUBC = C
            GOSUB ANGL1
```

```
File DB1:[ANNAL.BASE1.4] printed on 0/0/0

            RELBEG(SUBC  = DEG
            NEXT SUBB
            RELBEG(SUBD) = RELBEG(7) - RELBEG(3)
            SUBB = SUBB + 1
            READ A,B,C
            SUBA = A : SUBB = B : SUBC = C
            GOSUB PTLIN
            IF LM = 0 GOTO BEGI
            GOSUB CH12
BEGI :      RELBEG(SUBB) = LM
            GOTO REIDEL
REIDEL :    IF RID$ = "Y" GOTO REIDL1
            FOR SUBL = 1 TO 3
            READ A,B,C,D
            NEXT SUBL
            READ A,P
            GOTO PRTRT
REIDL1 :    FOR SUBL = 1 TO 2
            READ A,B,C,D
            SUBA = A : SUBB = B
            SUBC = C : SUBD = D
            GOSUB ANGL4
            RELRDL(SUBL) = DEG
            NEXT SUBL
            FOR SUBL = 3 TO 4
            READ A,B,C
            SUBA = A : SUBB = B : SUBC = C
            GOSUB ANGL3
            RELRDL(SUBL) = DEG
            NEXT SUBL
            RELRDL(SUBL) = RELRDL(3) - RELRDL(4)
            GOTO PRTRT
CH1 :       RETURN
CH4 :       IF XSTR(SUBA)<XSTR(SUBC) AND XSTR(SUBC)>XSTR(SUBD) DEG=DEG*(-1)
            IF XSTR(SUBA)>XSTR(SUBC) AND XSTR(SUBC)<XSTR(SUBD) DEG=DEG*(-1)
            IF XSTR(SUBA)=XSTR(SUBC) AND XSTR(SUBD)<XSTR(SUBC) DEG=DEG*(-1)
            RETURN
CH5 :       IF DEG = 0 RETURN
            CH5A = (YSTR(SUBA)-YSTR(SUBB)) / (XSTR(SUBA)-XSTR(SUBB))
            CH5B = (YSTR(SUBC)-YSTR(SUBD)) / (XSTR(SUBC)-XSTR(SUBD))
            IF XSTR(SUBA) < XSTR(SUBB) GOTO CH5RVR
            IF CH5A < CH5B AND CH5B < 0 DEG = DEG * (-1
            RETURN
CH5RVR :    IF CH5A < CH5B AND CH5B < 0 GOTO CH5TRN
            DEG = DEG * (-1)
CH5TRN :    RETURN
CH6 :       IF DEG = 0 RETURN
            CH6A = (YSTR(SUBB)-YSTR(SUBA)) / (XSTR(SUBB)-XSTR(SUBA))
            CH6B = (YSTR(SUBD)-YSTR(SUBC)) / (XSTR(SUBD)-XSTR(SUBC))
            IF YSTR(SUBB) < YSTR(SUBA) GOTO CH6RVR
            IF CH6A < CH6B AND CH6B < 0 DEG = DEG * (-1)
            RETURN
CH6RVR :    IF CH6A < CH6B AND CH6B < 0 GOTO CH6TRN
            DEG = DEG * (-1)
CH6TRN :    RETURN
CH10 :      DEG = 90 - DEG
            RETURN
CH11 :      DEG = DEG - 90
            RETURN
CH12 :      CH12A = (YSTR(SUBA)-YSTR(SUBB)) / (XSTR(SUBA)-XSTR(SUBB))
```

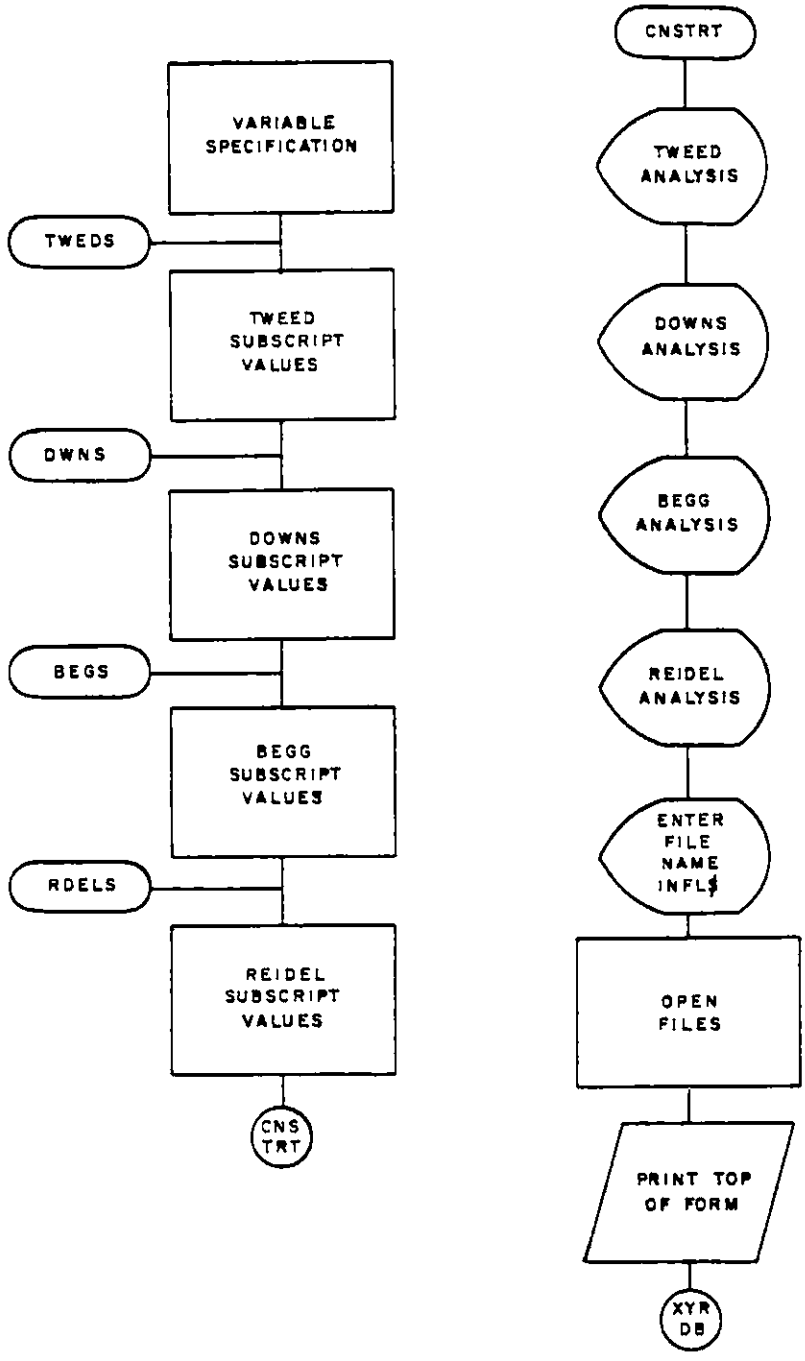
```
File DB1:CANNAL.BAS[1,4] printed on 0-0-0

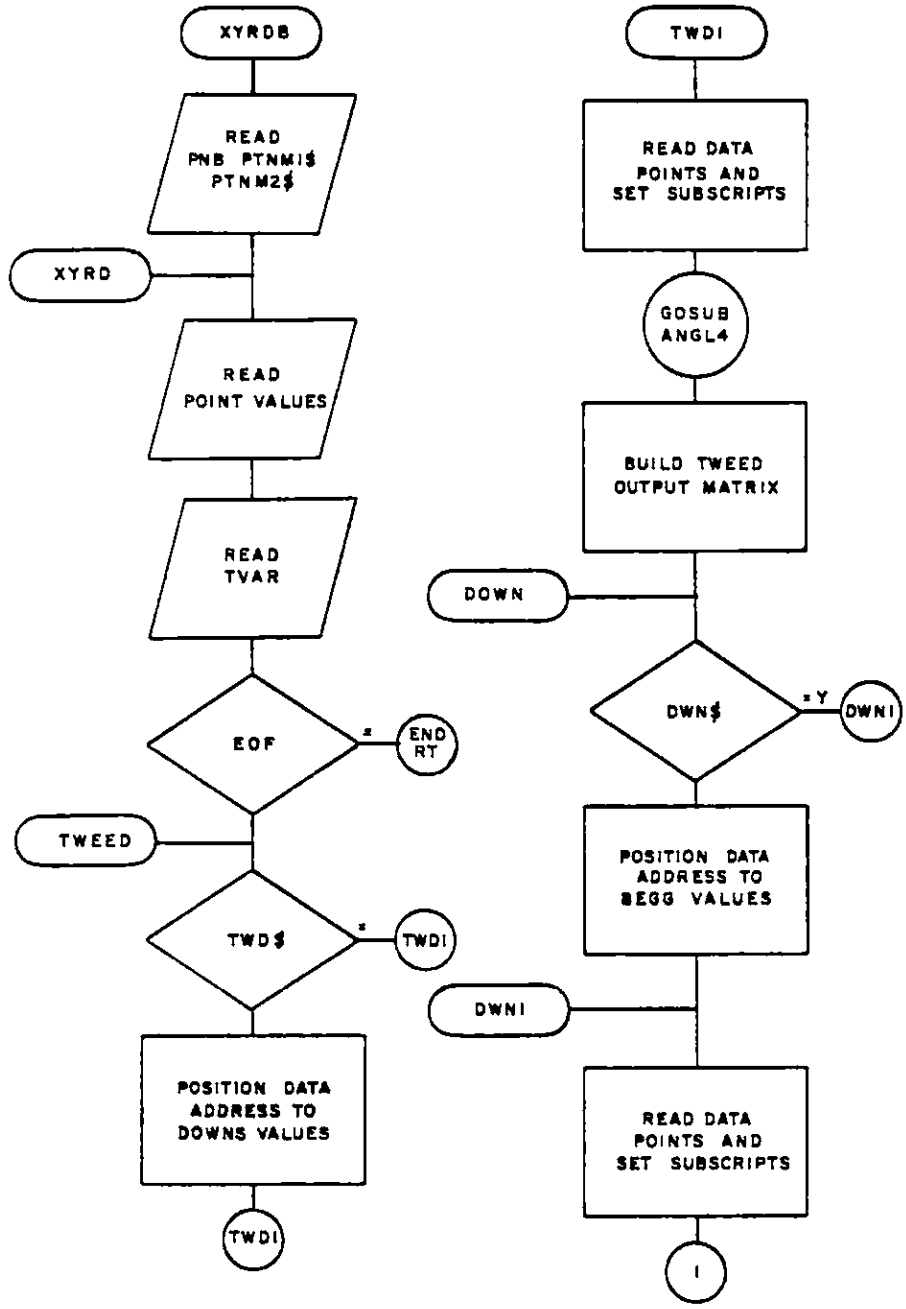
            CH12B = (YSTR(SUBC)-YSTR(SUBB)) / (XSTR(SUBC)-XSTR(SUBB))
            IF XSTR(SUBA) < XSTR(SUBB) GOTO CH12VR
            IF XSTR(SUBA) = XSTR(SUBB) GOTO STLN
            IF CH12A < CH12B AND CH12B > 0 GOTO CH12TRN
            LM = LM * (-1)
            RETURN
STLN :      IF XSTR(SUBC) < XSTR(SUBA) GOTO CH12TRN
            LM = LM * (-1)
CH12TRN :   RETURN
CH12VR :    IF CH12A < CH12B AND CH12B < 0 GOTO NEGCHG
            RETURN
NEGCHG :    LM = LM * (-1)
            RETURN
ANGL3 :     XASV = XSTR(SUBA) : YASV = YSTR(SUBA)
            XBSV = XSTR(SUBB) : YBSV = YSTR(SUBB)
            XCSV = XSTR(SUBC) : YCSV = YSTR(SUBC)
            DIFSV1 = SQR((XASV-XBSV)**2+(YASV-YBSV)**2)
            DIFSV2 = SQR((XBSV-XCSV)**2+(YBSV-YCSV)**2)
            DIFSV3 = SQR((XCSV-XASV)**2+(YCSV-YASV)**2)
            X = DIFSV1
            Y = DIFSV2
            Z = DIFSV3
            RAD1 = ACS((Z*Z)-(Y*Y)-(X*X))/-(2*Y*X))
            DEG = (RAD1 * 180)/3.1417
            RETURN
ANGL4 :     XASV = XSTR(SUBA) : YASV = YSTR(SUBA)
            XBSV = XSTR(SUBB) : YBSV = YSTR(SUBB)
            XCSV = XSTR(SUBC) : YCSV = YSTR(SUBC)
            XDSV = XSTR(SUBD) : YDSV = YSTR(SUBD)
            TXSV = XCSV-XASV
            TYSV = YCSV-YASV
            XDSV = XDSV-TXSV
            YDSV = YDSV-TYSV
            XCSV = XASV
            YCSV = YASV
            DIFSV1 = SQR((XCSV-XBSV)**2+(YCSV-YBSV)**2)
            DIFSV2 = SQR((XBSV-XDSV)**2+(YBSV-YDSV)**2)
            DIFSV3 = SQR((XDSV-XCSV)**2+(YDSV-YCSV)**2)
            X = DIFSV1
            Y = DIFSV2
            Z = DIFSV3
            RAD1 = ACS(((Y*Y)-(X*X)-(Z*Z))/-(2*Z*X))
            DEG = (RAD1 * 180)/3.1417
            RETURN
RTLIN :     XASV = XSTR(SUBA) : YASV = YSTR(SUBA)
            XBSV = XSTR(SUBB) : YBSV = YSTR(SUBB)
            XCSV = XSTR(SUBC) : YCSV = YSTR(SUBC)
            DIFSV1 = SQR((XASV-XBSV)**2+(YASV-YBSV)**2)
            DIFSV2 = SQR((XBSV-XCSV)**2+(YBSV-YCSV)**2)
            DIFSV3 = SQR((XCSV-XASV)**2+(YCSV-YASV)**2)
            X = DIFSV1
            Y = DIFSV2
            Z = DIFSV3
            RAD1 = ACS((Y*Y)-(Z*Z)-(X*X))/-(2*Z*X))
            H = Z * SIN(RAD1)
            L = H/200
            LM = (L/.3937) * 10
            RETURN
PRTRT :     PRINT #7," "
            PRINT #7,TAB(2);PNB;TAB(15);PTNM2$;TAB(32);TAB(34);PTNM1$
```

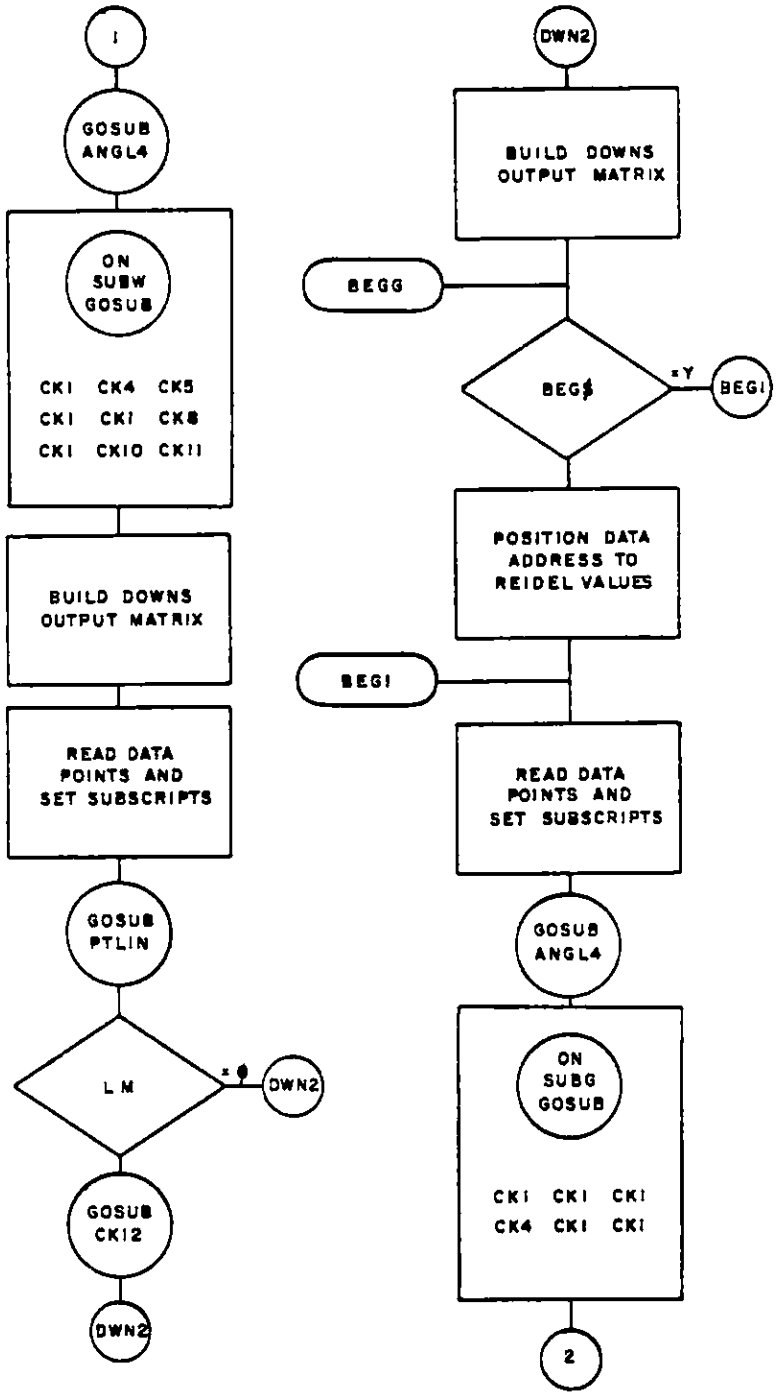
```
File DB1:CANNAL.BAS[1,4] printed on 2/2/2

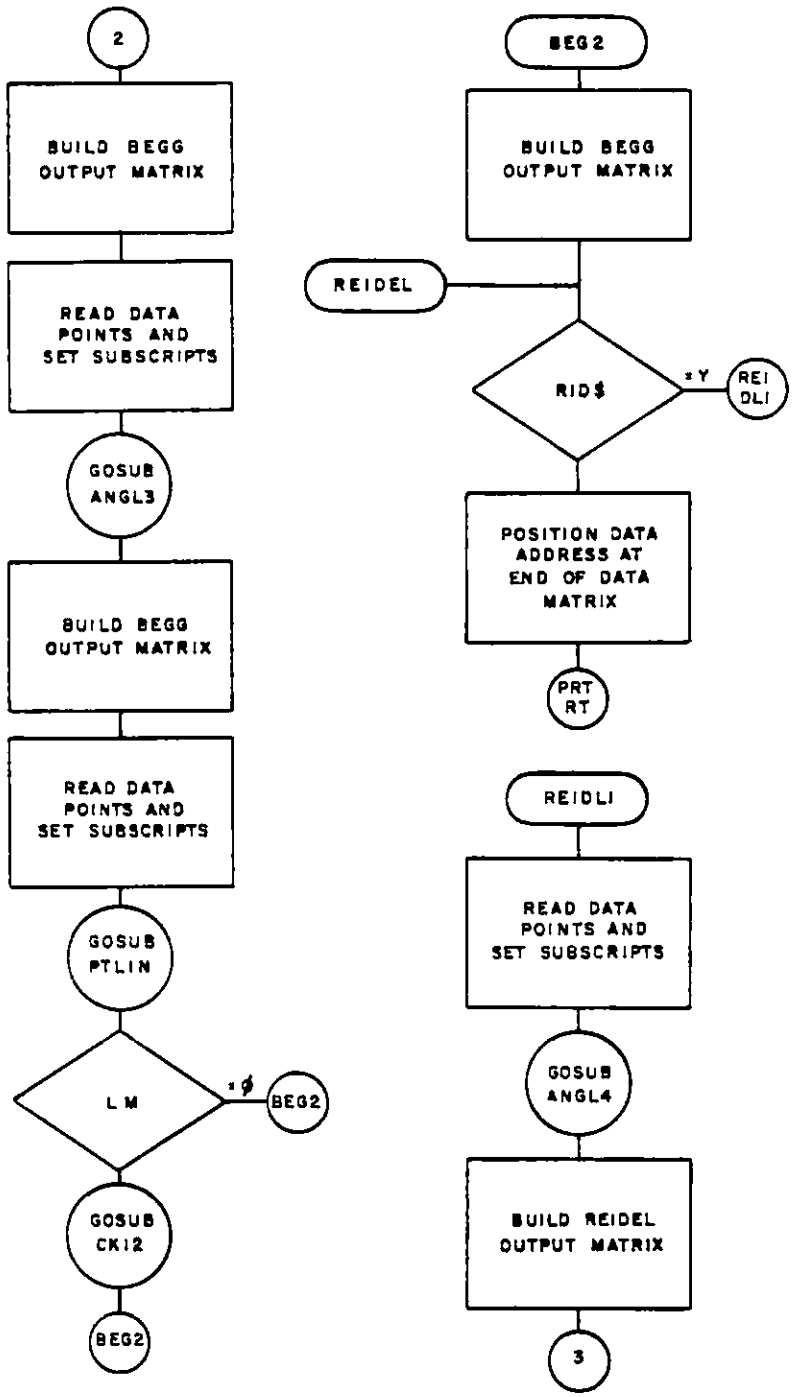
TWDPRT :    IF TWD$ <> "Y" GOTO DWNPRT
            PRINT #7," "
            PRINT #7,TAB(5);"TWEED ANALYSIS :"
            PRINT #7," "
            PRINT #7,TAB(10);"FMA. ";RSLTWD(1);TAB(30);"IMPA. ";RSLTWD(2);
            PRINT #7,TAB(50);"FMIA. ";RSLTWD(3)
DWNPRT :    IF DWN$ <> "Y" GOTO BEGPRT
            PRINT #7," "
            PRINT #7,TAB(5);"DOWNS ANALYSIS :"
            PRINT #7," "
            PRINT #7,TAB(10);"FFL. ";RSLDWN(1);TAB(30);"CONVX. ";RSLDWN(2);
            PRINT #7,TAB(50);"A.-B. ";RSLDWN(3);TAB(70);"MFL. ";RSLDWN(4);
            PRINT #7,TAB(90);"Y.-AX. ";RSLDWN(5)
            PRINT #7," "
            PRINT #7,TAB(10);"OFL. ";RSLDWN(6);TAB(30);"IN.INC. ";RSLDWN(7);
            PRINT #7,TAB(50);"L1.-OCC. ";RSLDWN(8);TAB(70);"L1.-MF. ";
            PRINT #7,RSLDWN(9);TAB(90);"U1.-AP. ";RSLDWN(10)
BEGPRT :    IF BEG$ <> "Y" GOTO RDLPRT
            PRINT #7," "
            PRINT #7,TAB(5);"BEGG ANALYSIS :"
            PRINT #7," "
            PRINT #7,TAB(10);"SN.MP. ";RSLBEG(1);TAB(30);"L1.-MP. ";
            PRINT #7,RSLBEG(2);TAB(50);"SN.-OPL. ";RSLBEG(3);TAB(70);
            PRINT #7,"CONVX. ";RSLBEG(4);TAB(90);"FFL. ";RSLBEG(5)
            PRINT #7," "
            PRINT #7,TAB(10);"U1.-SN. ";RSLBEG(6);TAB(30);"SNA. ";
            PRINT #7,RSLBEG(7);TAB(50);"SNB. ";RSLBEG(8);TAB(70);
            PRINT #7,"ANB. ";RSLBEG(9);TAB(90);"L1.-AF. ";RSLBEG(10)
RDLPRT :    IF RID$ <> "Y" GOTO CHRD
            PRINT #7," "
            PRINT #7,TAB(5);"REIDEL ANALYSIS :"
            PRINT #7," "
            PRINT #7,TAB(10);"L1.-FH. ";RSLRDL(1);TAB(30);"FH.-NA. ";
            PRINT #7,RSLRDL(2);TAB(50);"SNA. ";RSLRDL(3);TAB(70);
            PRINT #7,"SNB. ";RSLRDL(4);TAB(90);"ANB. ";RSLRDL(5)
            GOTO CHRD
CHRD :      RESTORE
            GOTO XYPDB
ENDPRT :    CLOSE #5
            CLOSE #7
            PRINT TAB(-1,0);TAB(10,10);"CANNAL HAS COMPLETED THE RUN."

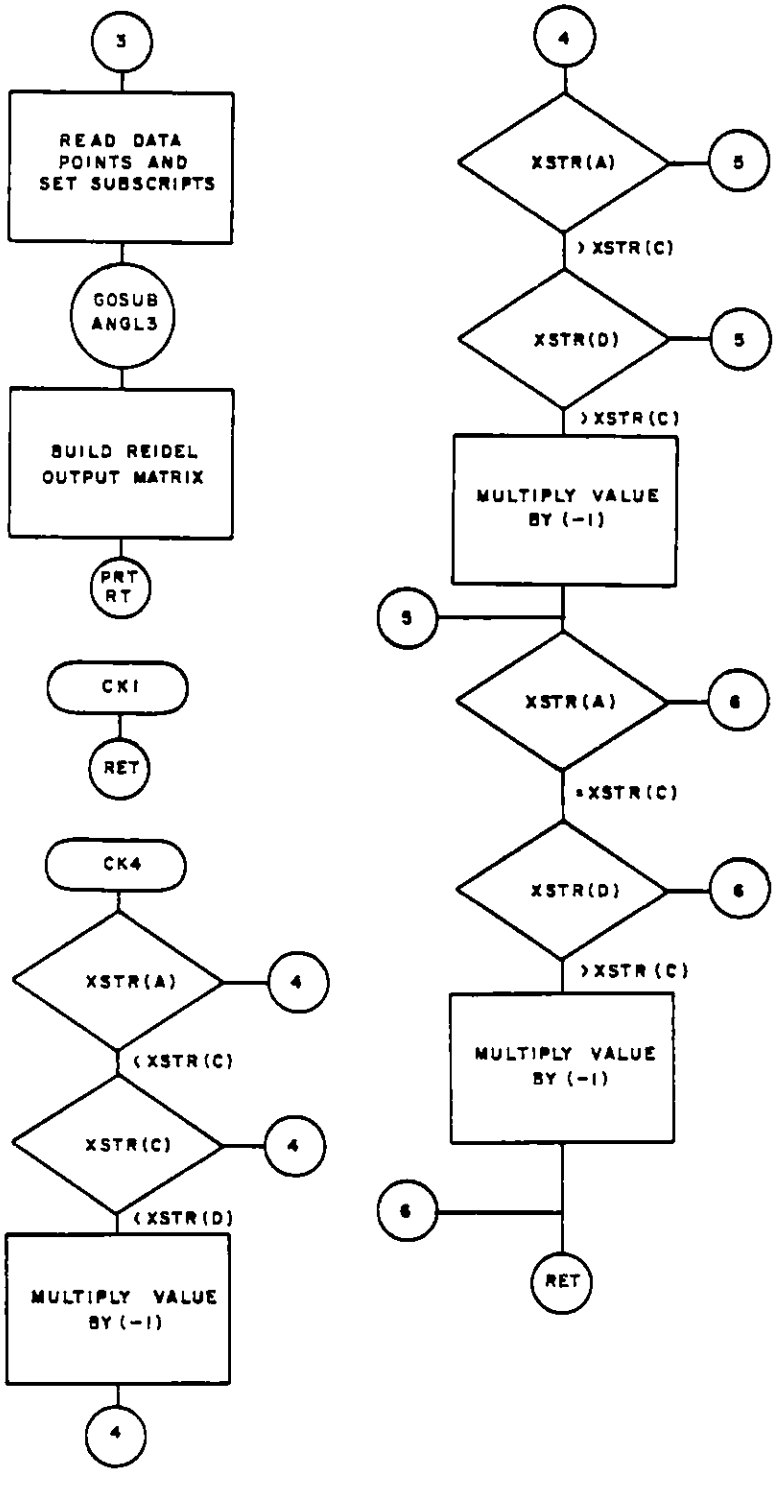
            END
```

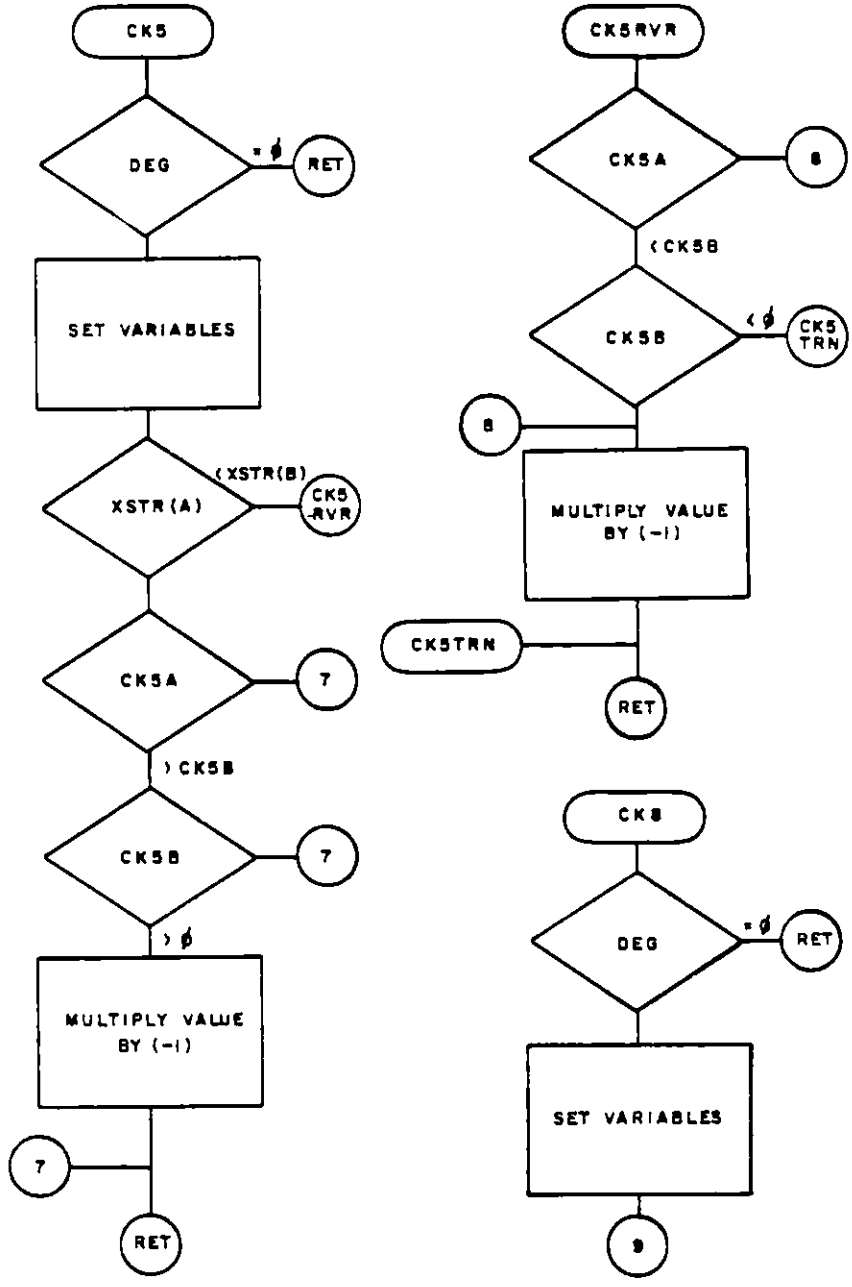

VARIABLE SPECIFICATION
TWEDS
TWEED SUBSCRIPT VALUES
DWNS
DOWNS SUBSCRIPT VALUES
BEGS
BEGG SUBSCRIPT VALUES
RDELS
REIDEL SUBSCRIPT VALUES
CNS TRT
CNSTRT
TWEED ANALYSIS
DOWNS ANALYSIS
BEGG ANALYSIS
REIDEL ANALYSIS
ENTER FILE NAME INFL$
OPEN FILES
PRINT TOP OF FORM
XYR DB

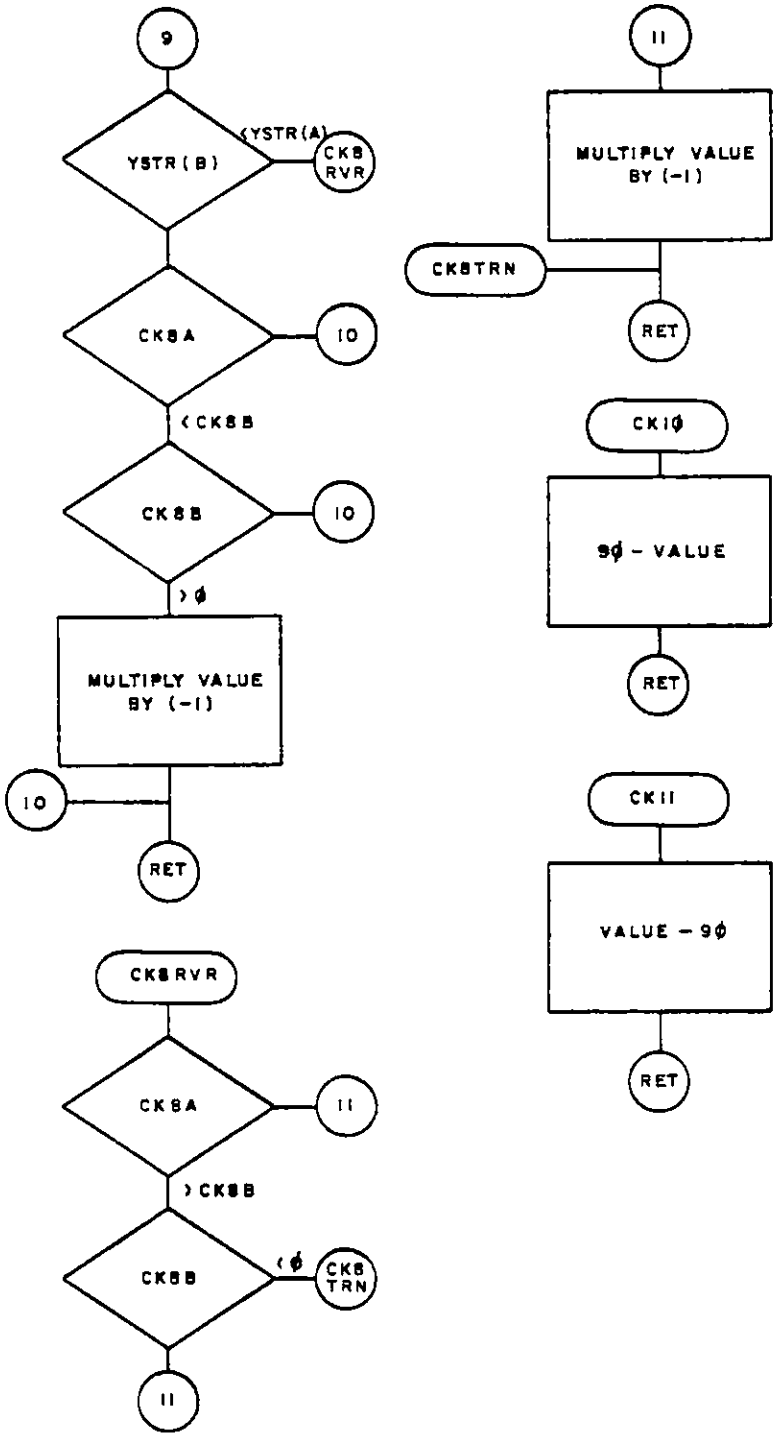

XYRDB
READ
PNB PTNM1$
PTNM2$
XYRD
READ
POINT VALUES
READ
TVAR
EOF
=
END
RT
TWEED
TWD$
=
TWDI
POSITION DATA
ADDRESS TO
DOWNS VALUES
TWDI
TWDI
READ DATA
POINTS AND
SET SUBSCRIPTS
GOSUB
ANGL4
BUILD TWEED
OUTPUT MATRIX
DOWN
DWN$
=Y
DWNI
POSITION DATA
ADDRESS TO
BEGG VALUES
DWNI
READ DATA
POINTS AND
SET SUBSCRIPTS
1

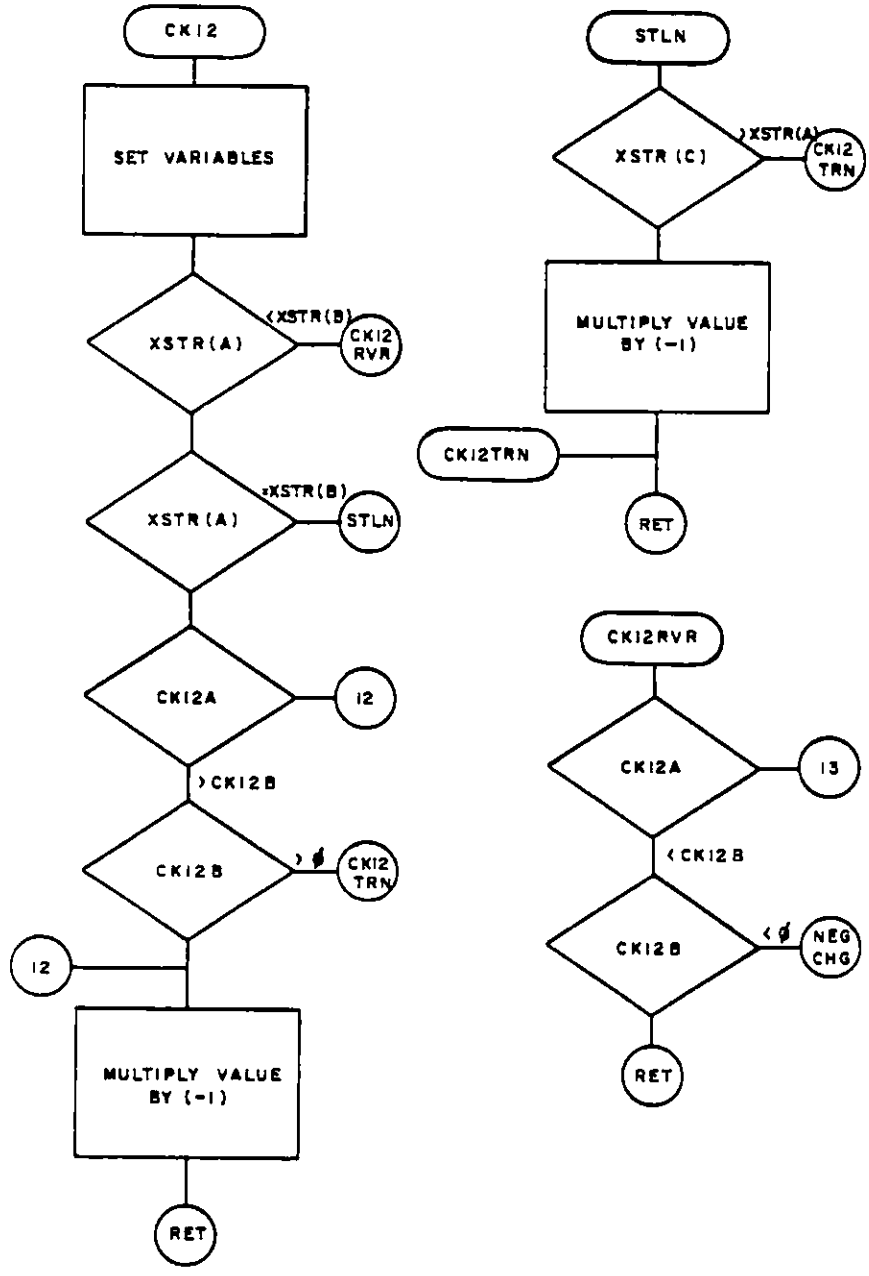

1
GOSUB ANGL4
ON SUBW GOSUB
CK1 CK4 CK5
CK1 CK7 CK8
CK1 CK10 CK11
BUILD DOWNS OUTPUT MATRIX
READ DATA POINTS AND SET SUBSCRIPTS
GOSUB PTLIN
L M
≠ 0
DWN2
GOSUB CK12
DWN2
DWN2
BUILD DOWNS OUTPUT MATRIX
BEGG
BEG$
≠ Y
BEG1
POSITION DATA ADDRESS TO REIDEL VALUES
BEG1
READ DATA POINTS AND SET SUBSCRIPTS
GOSUB ANGL4
ON SUBG GOSUB
CK1 CK1 CK1
CK4 CK1 CK1
2

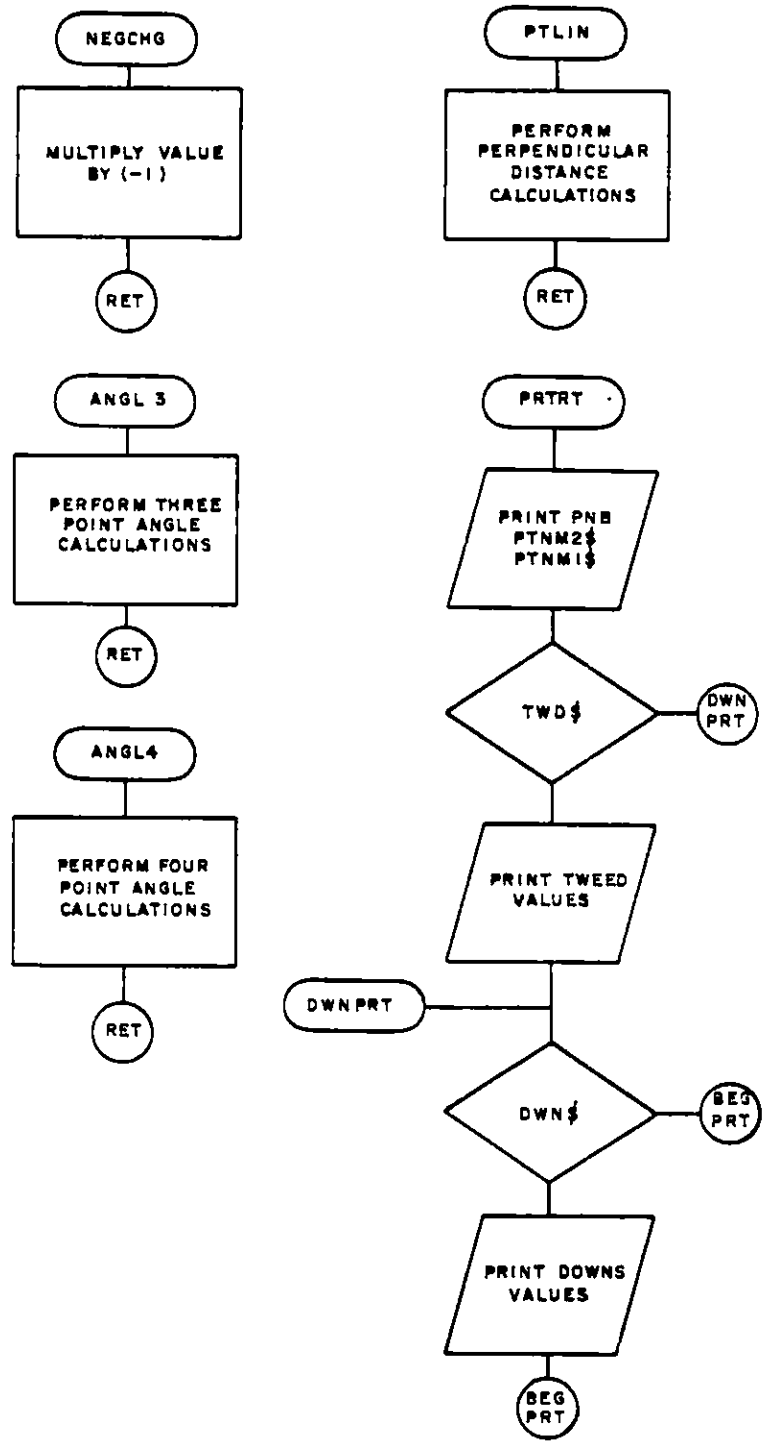

2
BUILD BEGG OUTPUT MATRIX
READ DATA POINTS AND SET SUBSCRIPTS
GOSUB ANGL3
BUILD BEGG OUTPUT MATRIX
READ DATA POINTS AND SET SUBSCRIPTS
GOSUB PTLIN
L M
≠ϕ
BEG2
GOSUB CK12
BEG2
BEG2
BUILD BEGG OUTPUT MATRIX
REIDEL
RID$
≠Y
REI DLI
POSITION DATA ADDRESS AT END OF DATA MATRIX
PRT RT
REIDLI
READ DATA POINTS AND SET SUBSCRIPTS
GOSUB ANGL4
BUILD REIDEL OUTPUT MATRIX
3

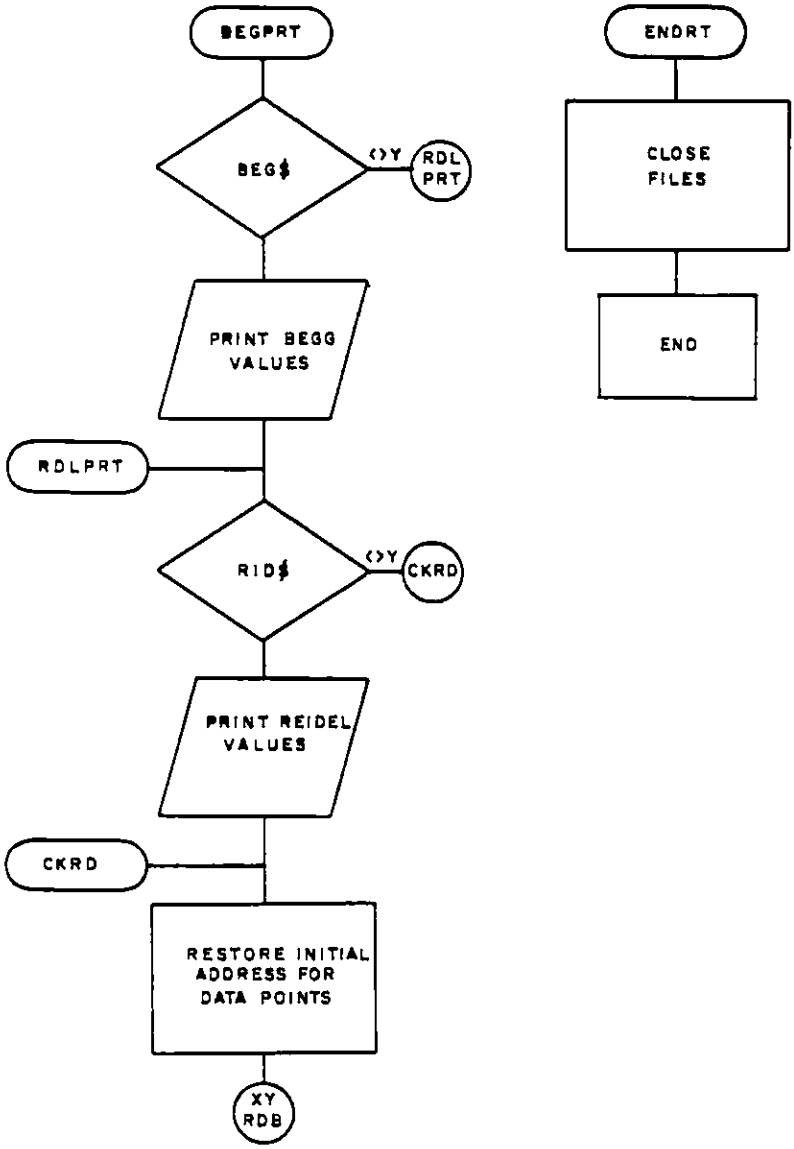

3
READ DATA POINTS AND SET SUBSCRIPTS
GOSUB ANGL3
BUILD REIDEL OUTPUT MATRIX
PRT RT
CK1
RET
CK4
XSTR(A)
4
(XSTR(C)
XSTR(C)
4
(XSTR(D)
MULTIPLY VALUE BY (-1)
4
4
XSTR(A)
5
)XSTR(C)
XSTR(D)
5
)XSTR(C)
MULTIPLY VALUE BY (-1)
5
XSTR(A)
6
*XSTR(C)
XSTR(D)
6
)XSTR(C)
MULTIPLY VALUE BY (-1)
6
RET


CK5
DEG
= ϕ
RET
SET VARIABLES
XSTR(A)
< XSTR(B)
CK5 -RVR
CK5A
7
> CK5B
CK5B
7
> ϕ
MULTIPLY VALUE BY (-1)
7
RET
CK5RVR
CK5A
8
< CK5B
CK5B
< ϕ
CK5 TRN
8
MULTIPLY VALUE BY (-1)
CK5TRN
RET
CK8
DEG
= ϕ
RET
SET VARIABLES
9


9
YSTR(B)
<YSTR(A)
CKS RVR
CKSA
10
<CKSB
CKSB
10
>ϕ
MULTIPLY VALUE BY (-1)
10
RET
CKSRVR
CKSA
11
>CKSB
CKSB
<ϕ
CKS TRN
11
11
MULTIPLY VALUE BY (-1)
CKSTRN
RET
CK1ϕ
9ϕ - VALUE
RET
CK11
VALUE - 9ϕ
RET


CK12
SET VARIABLES
XSTR(A)
<XSTR(B)
CK12 RVR
XSTR(A)
>XSTR(B)
STLN
CK12A
12
>CK12B
CK12B
>ϕ
CK12 TRN
12
MULTIPLY VALUE BY (-1)
RET
STLN
XSTR (C)
>XSTR(A)
CK12 TRN
MULTIPLY VALUE BY (-1)
CK12TRN
RET
CK12RVR
CK12A
13
<CK12B
CK12B
<ϕ
NEG CHG
RET


NEGCHG
MULTIPLY VALUE BY (-1)
RET
PTLIN
PERFORM PERPENDICULAR DISTANCE CALCULATIONS
RET
ANGL 3
PERFORM THREE POINT ANGLE CALCULATIONS
RET
ANGL4
PERFORM FOUR POINT ANGLE CALCULATIONS
RET
PRTRT
PRINT PNB PTNM2$ PTNM1$
TWD$
DWN PRT
PRINT TWEED VALUES
DWNPRT
DWN$
BEG PRT
PRINT DOWNS VALUES
BEG PRT


BEGPRT
BEG$
<>Y
RDL PRT
PRINT BEGG VALUES
RDLPRT
RID$
<>Y
CKRD
PRINT REIDEL VALUES
CKRD
RESTORE INITIAL ADDRESS FOR DATA POINTS
XY RDB
ENDRT
CLOSE FILES
END

APPENDIX X

SYSTEM BOOT PRINTOUT

```
 [
.MEMORY 4400
[4400 BYTES ASSIGNED]
.LOG 1,7
Logged into DSK0:[1,7]

.LPTINI PRINTR.INI
```

GENERAL BOOKBINDING CO
QUALITY CONTROL MARK