

201

A PLAN FOR THE INTEGRATION OF MICROCOMPUTERS  
INTO THE CIVIL ENGINEERING CURRICULUM  
AT KANSAS STATE UNIVERSITY

by

MICHELE C. PERRIN

B.S., Kansas State University, 1982

---

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

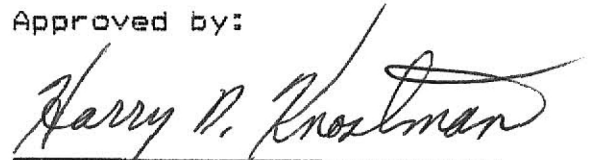
MASTER OF SCIENCE

Department of Civil Engineering

KANSAS STATE UNIVERSITY  
Manhattan, Kansas

1983

Approved by:



Major Professor

**THIS BOOK  
CONTAINS  
NUMEROUS PAGES  
WITH THE ORIGINAL  
PRINTING BEING  
SKEWED  
DIFFERENTLY FROM  
THE TOP OF THE  
PAGE TO THE  
BOTTOM.**

**THIS IS AS RECEIVED  
FROM THE  
CUSTOMER.**

LD  
2668  
.T4  
1983  
P47  
C.2  
Topic

A11202 578319

TABLE OF CONTENTS

Topic	Page
Purpose	1
Classification of Computers	2
Description of Microcomputers	4
Languages and Software	10
How Engineers Use Computers	13
Microcomputers in the Classroom	16
Academic and Social Responsibilities	19
Case Study No. 1	22
Case Study No. 2	24
Recommendations	26
References	32
Appendix I: An Introductory Course in Basic Programming for Civil Engineering Majors	36
Appendix II: Civil Engineering Apple Library Program Descriptions & User Instructions	70

## LIST OF TABLES

Topic	Page
Table 1: Microcomputer Specifications	8
Table 2: Microcomputer Specifications	9



## PURPOSE

The purpose of this paper is to research and present various ways that microcomputers are being used in the Civil Engineering profession. It is strongly felt by the writer that microcomputers are becoming an increasingly useful tool in numerical analysis and are becoming more popular in small engineering firms due to their low cost. Therefore, it is to the Civil Engineering Department's advantage to incorporate a strong program of microcomputer use into its curriculum in order to retain a high standard of educational excellence.

This paper seeks to pursue two aims. First, a thorough review is presented on microcomputer development, the types available, and how other institutions use microcomputers. Second, recommendations as to a program of implementation have been outlined. Included in this section are suggested equipment purchases, a sixteen week course outline in microcomputer programming, and instructions on operating various software programs developed by a Civil Engineering problems class. It is hoped that this analysis will help the faculty in making decisions concerning microcomputer incorporation into the classroom.

## CLASSIFICATION OF COMPUTERS

Until 1945, man's calculating speed was relatively constant depending upon his ability to handle the abacus and the slide rule. With the introduction of the first electronic computer, that calculating speed increased 100 times. Computer developments between 1951-1971 increased the speed 1000 times more. Presently, with the aid of computers, calculations can be performed in one-billionth of a second (nanosecond) and scientists are still striving for faster speeds.

Any type of instrument that aids a person in performing calculations can be classified as a computer. The abacus is a digital type of computer. Numerical manipulations are performed with a high degree of precision. A slide rule is an analog type of computer. It is usually faster than the abacus, because calculations are based on a series of approximations. Its accuracy, however, is dependent upon the human eye. The modern computer uses a combination of the digital and analog methods.

There are three major classifications of computers according to size. A mainframe is the largest type of computer with virtually unlimited storage capacity and processing capability. It is usually very expensive to buy and operate, costing in excess of \$100,000. A minicomputer contains at least 256K of memory. The cost ranges from \$20,000 - \$100,000 depending upon its memory size, processing speed, and auxilliary hardware. A microcomputer is the

cheapest of the three types. The memory capacity ranges from 16-256K with 48-64K as the typical size for most small office computers. It is the simplest to operate due to direct interaction between user and machine. The microcomputer can be used as a stand-alone processor where programs are written and run independently or it can be used as a terminal to a mainframe where programs written on the microcomputer are run on the larger machine.

The internal structure of a computer is based upon a switch. Flipping a simple switch on and off can represent many things: true and false, yes and no, one and zero, etc. When this switch is combined with other switches, any character, number, or letter can be represented using the binary system. Eight switches, called bits, are combined to form one character called a byte. One byte can represent a number as large as 255; two bytes are required for numbers up to 32,767. Decimal numbers always require at least two bytes. Bytes are measured in multiples of 1000 represented by a K or 1,000,000 represented by an M. Computer memory is usually measured in multiples of 16K (16,000 bytes or characters); therefore, the term 48K of memory represents the capacity to store  $3 \times 16,000 = 48,000$  characters.

There are two types of memory within any computer. Read Only Memory (ROM) is the memory used to store machine operations. Each button on the keyboard has been precoded into bits and bytes and stored in Read Only Memory. Common functions, such as SQR and EXP, and operating commands, such as RUN and STORE, are programmed within the machine so that

the user need only press the proper button to execute one of these operations. The programmer can use the information in this memory at any time, but can never store anything in it. Random Access Memory (RAM) is the memory reserved for user-defined programs and functions. Commands that are not inherent to the computer operating system can be coded and stored in Random Access Memory by the programmer. The information in RAM can be altered or manipulated at any time.

#### DESCRIPTION OF MICROCOMPUTERS

Microcomputers are a self-contained system. A standard set-up includes the microprocessor, keyboard, video monitor, printer, disk drive or cassette recorder, and auxilliary storage. The microprocessor is the "brain" of the microcomputer. All commands, calculations, and data manipulations are executed by this internal piece of hardware. The keyboard is the means of inputting information to the microprocessor. It is similar to a standard typewriter keyboard with the addition of a few extra keys for mathematical symbols. The video monitor and the printer are the means of outputting information from the microprocessor. The video monitor is similar to a television screen with the output displayed in black and white, green and black, or color. The green and black display is preferred over the black and white display when working on the microcomputer for long periods of time, because it is less irritating to the eyes. Printers range from dot matrix to letter quality with

many variations in between. The dot matrix printers output characters in a pattern of tiny dots, whereas letter quality printers use a typing element from a regular typewriter. Printing speed is dependent not only upon the brand of printer, but also upon how quickly the microprocessor can feed information to the print buffer. In general, dot matrix printers are about four times faster and one-third as expensive as letter quality printers. The disk drive or cassette recorder is the means of storing information from the microprocessor. The disk drive uses "floppy" disks as its method of auxiliary storage. The cassette recorder, while less expensive, is seldom used any longer due to its slow speed and awkwardness.

A microcomputer is very similar to a pocket programmable calculator. Both can code and store programs and print out results of calculations. They are inexpensive, lightweight, and relatively portable. The microcomputer, however, has many advantages over the programmable calculator. Much larger programs can be stored and run and calculations can be executed more rapidly. Microcomputer programs can be edited or modified more easily than calculator programs. The printout from a microcomputer is suitable to use directly in an engineering report. The output can be printed on 8-1/2" X 11" paper in a structured format with suitable headings and labels, whereas the printout from a calculator is usually a string of numbers printed on a paper tape.

Microcomputers were designed to be used by one individual at a time. They can be used independently with their own

microprocessor or as a terminal to the central processing unit of a large mainframe. They do not function well if multiple terminals (keyboard and video monitor) are connected to one microprocessing unit. Multiple users increase processing time and reduce available memory capacity. If one user is doing especially heavy work or tying up an auxilliary part of the computer (such as the printer) for any long period of time, the other users must wait until he is finished. Minicomputers were designed specifically to support several users at one time. Radio Shack has developed a TRS80 Model 16 which contains 128K memory and will support three additional terminals. Programmers must use a "hard" disk as opposed to the floppy disk. The hard disk, similar to an LP record album, is more expensive and less portable than the floppy disk, but it provides a more durable form of software storage.

Some features of the more common brands of microcomputers have been summarized in Tables 1 and 2. The KSU Civil Engineering Department currently possesses three Apple II and three TRS80 Model II microcomputers. After two years of operation, many advantages and disadvantages can be seen in both types of microcomputers. The TRS80 seems to be the more durable of the two machines. There have been fewer breakdowns and the keyboard has retained a better touch sensitivity. The Apple II, however, appears to be the machine favored by faculty and graduate students when programming application software, due to its ease of operation and graphics capabilities. Programs can be stored on the Apple disk in such a manner that no machine commands are required to run the

program. A user merely inserts the disk and turns on the machine to begin operation. In addition, programs can be highlighted graphically to demonstrate principles and cut down on lengthy explanations. In a Civil Engineering curriculum, this feature can be particularly beneficial in such areas as structural analysis, influence lines, elastic curves, and shear and moment diagrams.

The TRS80 contains an 80-column video display as opposed to the 40-column Apple display. Opinion is mixed as to which display is preferable. The Apple uses a 5-1/4" disk which is less expensive than the 8" TRS80 disk, but has a smaller storage capacity. (Newer models of the TRS80 are switching to the 5-1/4" disk.) The TRS80 microcomputer has the unique capacity to relay the contents of the video display to the printer at any time during machine operation. This feature is especially useful when the programmer has been working in immediate mode and wishes to have a hard copy of the results.

TABLE 1: MICROCOMPUTER SPECIFICATIONS

Features	TI		
	Apple II+	Apple IIe	Professional
Microprocessor	Z80	Z80	8088
RAM	48K	64K	64K
Expandable Memory	yes	yes	yes
Display (col X lines)	40 X 24	40 X 24	80 X 25
Upper Case	yes	yes	yes
Lower Case	no	yes	yes
Typewriter Keypad	yes	yes	yes
Numeric Keypad	no	no	yes
Color Graphics	yes	yes	yes
Resolution	280 X 192	280 X 192	720 X 300
Disk Size	5-1/4"	5-1/4"	5-1/4"
BASIC	std	std	std
COBOL	extra	extra	extra
FORTRAN	extra	extra	extra
PASCAL	extra	extra	extra



TABLE 2: MICROCOMPUTER SPECIFICATIONS

Features	TRS 80		
	IBM	Commodore	Model IV
Microprocessor	8088	6510	Z80
RAM	64K	64K	64K
Expandable Memory	yes	yes	yes
Display (col X lines)	80 X 25	40 X 25	80 X 24
Upper Case	yes	yes	yes
Lower Case	yes	yes	yes
Typewriter Keypad	yes	yes	yes
Numeric Keypad	yes	no	yes
Color Graphics	yes	yes	yes
Resolution	320 X 200	320 X 200	640 X 240
Disk Size	5-1/4"	5-1/4"	5-1/4"
BASIC	std	std	std
COBOL	extra	no	extra
FORTRAN	extra	no	extra
PASCAL	extra	extra	extra

## LANGUAGES AND SOFTWARE

Engineering software for use on the mainframe computers was commonly written in Fortran. This computer language was capable of manipulating large quantities of data and outputting the results into a concise, tabular format. Programs on the microcomputer, however, are more often written in Basic. Basic, which stands for Beginners' All-purpose Symbolic Instruction Code, is an interactive language. Data entry, calculations, and output are performed very easily and quickly. Because of the limited memory capacity on a microcomputer, there is no noticeable difference between the calculating speed of a Fortran program versus the speed of a Basic program.

Most of the old Fortran programs written for the mainframe were too large to run on a microcomputer. Since mainframe operation was so expensive, programmers often wrote software that would perform a multitude of operations in one run. Microcomputer operation, however, requires efficient memory usage. Programs had to be broken down into individual operations. However, it was soon discovered that even with fewer requirements, the Fortran programs were using too much memory space. Fortran is not a very conservative language. For example, Fortran will not allow the use of variables to size a matrix, thus the matrix must be set sufficiently large enough when the program is written to cover all sizes of problems. On the other hand, using Basic, matrix dimensions can be sized each time a program is run.

Another advantage of Basic over Fortran relates to the method of data input and output. Fortran is based on the premise of a large quantity of data being fed into the computer, a series of calculations being performed on that data, and all results being printed out into neat, structured tables. Basic is much more interactive. It allows the user to enter pieces of data at different points within a program. The user can make decisions on subsequent operations based upon the results of intermediate calculations. At any time, data can be changed, added, or relayed to the printer for a paper copy. This type of question and answer interaction can be done very easily using Basic, but is very time consuming using Fortran.

Fortran and Basic fall into the category of "high-level" languages. These languages are fairly easy to learn and are relatively compatible between different types of computers. Coding in Basic on the TRS80 is almost identical to coding in Basic on the Apple. Each machine uses essentially the same commands set up in the same structure. There are some direct commands which are unique to each machine, but the same operation can almost always be performed in some way on other machines. For example, numerical data can be output to any number of significant figures through a "PRINT USING" command on the TRS80. Significant figures are determined on the Apple using a somewhat more complicated subroutine.

Pascal is another high-level language similar to Fortran. It uses the "stacking" principle to execute programs, a process similar to that used on an HP calculator. Assembler

or "machine" language is the most efficient method of programming since the user communicates directly with the microprocessor. This is the fastest processing method available, but requires a very skilled programmer to code properly. Many parts of the operating system, stored in Read Only Memory, are usually written in Assembler. Machine language, while powerful, is unique to each microprocessor and is not compatible with different brands of computers. Pilot and Logo are "end-user" languages which allow a non-data processing person to program a computer. These languages were developed to use with smaller home computers and to teach beginning programming skills to elementary school students. They are a fairly new concept and still in the experimental stage.

Many types of prepackaged programs have been written for use on the microcomputers. Word processing packages, such as Scripsit and Applewriter, create, store, and print out written text. Some include an editing feature which corrects spelling errors. Accounting packages, such as Visicalc and Multiplan, produce multicolumn account sheets used in budgeting, finance, and numerical analysis. Financial management packages, such as the Weekly Planner or Dow Jones Market Analyzer, can be used as aids in balancing checkbooks or tracking investments.

Currently, there is a severe lack of usable microcomputer software available to Civil Engineers. It has only been during the past two years that any attempt has been made to develop software for Civil engineering applications.

## HOW ENGINEERS USE COMPUTERS

The invention of computers provided engineers with the ability to solve problems that would have taken a lifetime to solve by hand. Early computers, however, were very expensive and ownership was limited to larger corporations. Smaller firms wanted to become more machine oriented in order to increase their efficiency, but could not afford to purchase a computer. A plan of "time sharing" was developed whereby a smaller company installed a remote terminal in its office and purchased computer time from a larger firm. This alternative, while faster than doing calculations by hand, was very expensive and not nearly as efficient as having an in-house computer available.

As computer development became more refined, the cost, as well as size, began to come down. Microcomputers are now affordable for small businesses, but it was recently estimated that only 20% of the engineering firms in the United States have computer facilities within their company. It is predicted, however, that during the next five years there will be an exponential growth of microcomputers in small and medium firms due to Civil Engineering application programs becoming more available. Today's students will probably go to work for companies who have just purchased or are thinking of purchasing microcomputers. Therefore, if they have had microcomputer exposure in college, they will be in a better position to be a more productive and useful employee to their company.

An in-house computer allows a firm to adapt the system, language, and hardware arrangement to suit its particular needs. Repetitious mathematical functions are quickly executed on a microcomputer. More complicated functions such as quadratic equations, simultaneous equations, or area under a curve are easily programmed into the microcomputer memory. Microcomputers are readily adapted to iterative processes such as structural analysis or hydraulic flow. Within the engineering firm, microcomputers have been used for job cost accounting, payroll, and word processing. The main product of a consulting firm is its reports, therefore it is vital that they be attractive, project a good image, and be produced as efficiently as possible.

Microcomputers can aid engineers in conventional as well as unconventional experiments. Simulation of a mathematical model is especially successful where basic physical laws are well understood. The models must be combined in extremely complicated ways in order to solve real problems and microcomputers can effectively coordinate all aspects of the simulation. For example, microcomputers are effective in the design of structural frames. An engineer selects column and beam sizes based on past experience, then runs a structural analysis to locate any excess deflections or stresses under different loading conditions. Critical members are resized and the entire analysis run again. In the past, this process has been done by hand, but with a microcomputer many more iterations can be handled thus developing a more optimal design.

This same technique has been successful in reducing building energy use. Projects of the future must not only be safe, cost-effective, and environmentally sound, but must also be resource efficient and technologically appropriate. Building orientation, glass areas, and levels of insulation in walls and roof were previously "one-shot guesstimates". By simulating wind loads on different faces of the building, the engineer can select the optimum site location.

Microcomputers can improve a consulting firm's customer relations. After a structure is built, if a client wants to put a six-ton load on one of the floors, a quick analysis can be run determining the effect of this additional load. Using a microcomputer, the consulting firm can produce an answer in about an hour; the same calculations would take about two days by hand.

Consulting firms are also incorporating a more specialized form of microcomputer. Computer-aided design and drafting systems are increasing the design possibilities of a consulting firm as well as producing more accurate and higher quality drawings. With the cost of labor so high, it is hard for a consulting firm to be competitive when it spends days producing a good quality drawing. Many plotters today, can generate a final drawing (ink on vellum) in about ten minutes. Interactive CAD systems can be useful in the preliminary design stages. As the design is developed, it can be continually checked by showing perspective views from all angles. CAD can also be used in conjunction with the STRUDL software package to check the structural effectiveness. All

these advantages help to produce a better quality project.

#### MICROCOMPUTERS IN THE CLASSROOM

School is no longer a place where a person accumulates most of the knowledge he will need over a lifetime. For one thing, most of the knowledge he will need has not been discovered yet; for another, much of what is now being taught, will be obsolete or irrelevant in a few years. Students need to develop a greater ability to understand and handle situations as they arise. Education often becomes a memorization habit with no clear understanding of principles. Knowledge of computer manipulation is the safest investment an engineer can make toward his future.

Using computers to solve problems imposes a strong discipline on the student by forcing him to analyze a problem in a logically consistent manner. Microcomputers can increase classroom effectiveness by simulating "hands-on" experiences. In the past, teachers could not give many real life problems to students because of the tedious and time consuming calculations involved. Simulating situations on a microcomputer can help a student gain insight into a problem by making it possible for him to see the consequences of his or other's actions in a short amount of time. For example, in a Freshman design course at the University of Missouri - Rolla, students were asked to redesign the bow of a bow and arrow. Their goal was to store the maximum amount of



mechanical energy with a mechanism that had the minimum amount of inertia. Normally, the students would have been limited in their design by the availability of materials and the method of construction. However, they used a microcomputer simulation program, which took three hours to write by the instructor, to test their designs.

Microcomputers are proving very valuable in laboratory courses. Since students are trying to mix newly learned problem solving skills with newly learned laboratory skills, an error in one usually results in an error in the other. By learning to evaluate data on a microcomputer system, emphasis shifts to running and analyzing the experiment. Also, since it has been found that student interest appears to be inversely proportional to the time required to obtain results, immediate feedback on the data analysis while still in the laboratory would enhance a student's comprehension of the experimental concept.

Students should be able to use microcomputers as problem solving aids, design and graphics display aids, data taking and control devices, and records management. Courses, such as Matrix Algebra, should be given greater importance in the curriculum since the most efficient computer analyses are based on matrix manipulation. All faculty generated programs for classroom use should follow a specified format. The best success occurs if the following sections have been identified and explained: program objective, program variables, variable format, special instructions, and program statements. Thorough documentation should explain key areas of the program

and eliminate user difficulties. The purpose of these faculty generated programs should be classroom demonstration or initial problem solving assignments. Later, upperclassmen can learn how to modify these programs to solve more advanced types of problems.

It should be recognized that part of the difficulty with students mastering microcomputers is due to inadequate indoctrination. Most students are unfamiliar with interactive programming and terminal use. If a student is continually frustrated with machine operation such as how to turn the machine on, how to load a program, or how to output the data, the whole purpose of the microcomputer assignment will be wasted. Many professors have ruined the effectiveness of their endeavors by neglecting this crucial aspect.

The emphasis on microcomputers in education should be on students learning to use the computer and not on students learning to use a computer program. Packaged software programs should be only a part of a student's computer education and should be used with care. Students develop a certain amount of passivity when everything is done for them. The student should learn there are a variety of ways a microcomputer can be of service to the Engineering profession and the more confident he is in microcomputer use, the better engineer he will be.

## ACADEMIC AND SOCIAL RESPONSIBILITIES

Computers of all types are definately exerting a strong influence on the course of engineering technology. Currently, however, this influence is largely unregulated and little understood. Microcomputers, if used effectively, can open up design options previously relegated to science fiction novels. But this increase in technology must alter our priorities and focus more strongly on the balance between an engineer's creativity and his social responsibility.

Many of the first software programs such as ICES, STRUDL, and NASTRAN required tremendous memory and mass storage, were not documented well, were slow to run, and harbored a small degree of uncertainty as to their use. Program errors often arise from trying to do too many things in one program. A program might have initially been written for one purpose, but is modified as time goes on to serve other purposes. The program may run well for a long time until a certain set of data tend to make it function improperly. The grave consequence is that many people are not sufficiently familiar with the algorithms used to solve problems to recognize an incorrect solution if the computer generated one.

Obviously, an engineering firm would not have the time or resources to write all their own software to insure its credibility. However, engineers involved with using computers should be sufficiently skilled to be able to read and understand a computer program just as a construction foreman should be able to read a set of blueprints. Universities are

the natural place for this type of training to begin, but there, too, computer use has been largely unregulated and left to the whims of the staff.

There are some good, as well as bad, examples of microcomputer use in higher education. Johns Hopkins uses interactive microcomputer drills to facilitate math and science courses. Since the student is acting on a one-to-one basis with the microcomputer, he is allowed to proceed at his own pace. Rensselaer Polytechnic Institute has become a leader in computer graphics. They are getting away from the common sense and intuitive approach to problem solving and, instead, are letting the students experiment and discover for themselves what makes a good solution. Arizona State University uses packaged software programs to let students explore multiple design options without having to perform the repetitive calculations.

All of these situations are commendable and definately have their place in the classroom. However, educators may be overlooking one dangerous aspect of this situation. If the students are not taught some form of computer programming at the same time, they must accept the reliability of these programs completely on faith. A program is only as reliable as the person who developed it. All aspects of computing must be introduced and none slighted or de-emphasized in any way. Just showing students the equipment in a casual introductory manner does not prepare them for future decision-making roles. Having access and the capability to run the entire system gives an overall knowledge that is learned faster and gives

more versatility.

Three important aspects relating to microcomputers should be stressed throughout a student's career. First, the prospective engineer should be familiar enough with programming language that he can verify a program's authenticity. A thorough knowledge in this area would assist him in making necessary modifications to an existing program to perform different functions. Without this knowledge, it would be very easy for engineers to misapply a computer program and/or extend themselves beyond their area of expertise. Second, an engineer should be able to use a microcomputer interactively as a calculating tool without the aid of formal programs. This aspect of a microcomputer's usefulness is often overlooked or unrecognized. A microcomputer is faster and simpler to use than a handheld calculator and has infinitely more storage capacity. Third, and most important, an engineer should be able to recognize, or at least suspect, a questionable solution. This is vital to the future well-being of an engineer's credibility. Even if the engineer has no part in the actual programming or running of the microcomputer analysis, if he is handed a computer solution to use in his job, he should feel confident that that solution is an accurate one.

CASE STUDY NO. 1

A microcomputer course was introduced into the Construction Science and Architectural Engineering curriculum two years ago. The three-hour course entitled "Introduction to Construction Programming" is required for all Freshman and Sophomore students. Emphasis is on Fortran programming.

The course content is structured to include microcomputer operation (one week), Fortran programming (eight weeks), computer graphics (two weeks), programmable calculators (two weeks), and computer applications (two weeks). The majority of the course is taught using four TRSBO Model II's; however, the computer graphics portion of the course is run on the IBM central mainframe computer. The microcomputers are located in an unused faculty office. Each work station includes one microprocessor, one keyboard, one video monitor, two disk drives, and one dot matrix printer. The computer room is open from 7:00am to 5:00pm Monday thru Friday. The close proximity to other faculty offices provides adequate supervision and security of the equipment.

The course is divided into two-one hour lectures and one-two hour laboratory (two different lab times are available per week). Lecture time is devoted to computer structure and language development. Homework problems are assigned during the laboratory session (one assignment per week). The problem is explained, an overall outline is worked through, and programming hints are given. Students may work on the homework assignments any time during the week. Two professors

are usually available to help with problems. Approximately 60-70 students are enrolled in the class each semester. There is no textbook for the course, but each student buys a pre-initialized disk from the department at the beginning of the semester. Programming information comes from lecture notes and class handouts.

The class appears to run very smoothly, especially for the large number of students enrolled. The biggest problems occur at the beginning of the semester when students are first learning to use the equipment. The first assignment is purposefully kept simple in order to strengthen operating skills. Also, since the faculty pre-initialize the disks, students do not need to become involved with the disk operating system until later in the course. There does not seem to be a problem with students getting the assignments done within a week's time. There are slack times on computer usage when a problem is first assigned and backlogs when the problem is due, but students quickly learn to budget their time throughout the week.

A TRS80 Model 16 was purchased for faculty and staff word processing and programming. This machine contains over five times the memory capacity of the Model II's and can accommodate additional terminals (keyboards and video monitors). Students are not allowed to perform any word processing tasks on any of the machines.

In summary, the department feels it has made a good investment. The equipment is very durable and there has been little problem with breakdowns. The second disk drive appears

to be the only unnecessary purchase since it is used only once a semester when the faculty initialize the student disks.

## CASE STUDY NO. 2

Through a series of NSF grants, the faculty at Wichita State University have successfully incorporated microcomputers into their engineering curriculum. The main emphasis of their program is to use microcomputers as a demonstration tool in the classroom to help explain engineering principles.

WSU currently possesses ten Apple II microcomputers and five projectors and large view screens. All the equipment is stored on portable carts in a central media resource center so that it can be set up as required in classrooms and lecture halls. The 3' X 5' viewing screens were purchased from Kloss Manufacturing at a cost of \$400 each. One screen can easily be seen by all students in a class size of approximately 70 students. The Nova Beam projectors cost \$2000. They are able to enlarge and transfer the video display as it appears on the microcomputer to the viewing screens.

All software used for classroom demonstrations has been written by faculty members. When the NSF grants were submitted, it was stipulated that monetary support would be reserved for summer faculty salaries. In this way, the faculty can devote full time to software development without classtime interruptions. A group of approximately 4-5 faculty work on software development each summer. Each person works



independently on his or her own program with weekly group meetings to discuss problems and solutions.

Many of the programs recently developed use a computer simulation in place of an analytical model to explain an engineering concept. The dynamic graphical display helps students to visualize relationships as they are introduced. It also has the advantage of increasing student interest by varying the style of classroom presentation.

For example, a queueing simulation program demonstrates the relationship of customer service rates at drive-up windows. A random generator approximates customer arrivals so that students can observe and analyze such factors as the average waiting time in the queue, the average service rate, or the number of customers denied service due to an overloaded system. A coin flip simulation program demonstrates the relationship between the binomial and geometric statistical distributions. A professor could not physically flip 1000 coins, but the microcomputer can simulate 1000 flips of a coin in approximately 60 seconds. At the same time, the student can watch the distributions, drawn in a histogram format, "grow" as the microcomputer progresses. The student can actually see how and why the various probability distributions were developed.

Overall, the faculty at WSU are very pleased with the impact the microcomputer demonstrations are having on student comprehension of material. Their findings have been supported by IDEA and LASTIC standardized instructor/instruction evaluations.

## RECOMMENDATIONS

It has been shown that microcomputer use is becoming more widespread among Civil Engineers. They are being substituted as an analytical tool for the more expensive mainframe computers and as a computational tool for the limited memory desktop calculators. Therefore, it is imperative that prospective engineers learn to manipulate microcomputers while they are still in college.

Three areas of emphasis are recommended when incorporating microcomputers into the Civil Engineering curriculum. First, an introductory programming course should be taught within the Civil Engineering Department using microcomputers as the main programming media. Second, simple software packages should be incorporated into laboratory classes to reduce repetitive calculations. Third, demonstration programs highlighted with graphics and sound should be used during classroom lectures to enhance a student's understanding of theoretical concepts.

The microcomputer programming course should be a Freshman/Sophomore level requirement, replacing the mainframe course currently being taught. It is felt that there will be no disadvantage in eliminating any use of the mainframe from the curriculum, since the only difference between running a program on a microcomputer versus running the same program on the mainframe is in the physical operation of the two machines. The emphasis should be on the Basic programming

language. It is easy to learn and commonly used by many Engineering firms. In addition, it appears to be the most interactive language in allowing the user to manipulate part or all of the data anywhere within the program. Emphasis should also be on immediate mode programming where the student uses the microcomputer much like a calculator to solve complex, iterative problems. In this mode of microcomputer use, no formal program is written and stored in the microcomputer. Rather, the student enters commands directly into the machine as they are needed.

A thorough knowledge of microcomputer usage and programming early in a student's curriculum will facilitate the incorporation of microcomputers into the classroom by faculty. Graphics programs can be developed to demonstrate shear and moment diagrams, deflection curves, influence lines, etc. Simple "what if" experiments can be executed very rapidly allowing students to see the effects of different values. Randomly generated situations can approximate true life occurrences, such as rainfall data.

Simple software programs incorporated into a laboratory or recitation class would allow a student to analyze the results of an experiment before leaving the laboratory. For example, students learn the basic stress and strain equations in Mechanics of Materials lecture. There is a laboratory experiment where they load various rods to failure, calculating the stresses at even load increments and plotting a curve. These calculations, while time consuming when done by hand, can be executed very rapidly on a microcomputer

allowing students to complete the laboratory assignment during the lab time. If an error was made in the experiment, it can be detected immediately and the experiment redone. This technique is currently being used in the Surveying classes. After the students run an outdoor survey, they enter the data into a traverse program. The program calculates bearings, azimuths, and area. However, if the student has made a serious error, it is detected immediately and the survey rerun.

To facilitate these goals, the department should set up a computer users area. It is recommended that two separate areas be designated: one for undergraduate student use and the other for faculty and graduate student use. The undergraduate microcomputer room should be located in a fairly sheltered area so that it can be kept unlocked and accessible to students from 8:00am to 5:00pm Monday thru Friday. Initially, the room should house four microcomputers, four disk drives, four video monitors, three dot matrix printers, and one letter quality printer. The TRS80 Model II's are suggested for this purpose, since they are lower in price than many of the other microcomputers and appear to be very durable. Keys to the microcomputer room can be checked out for computer use after 5:00pm. However, there should be a much stricter check out policy than is currently being used to guard against equipment misuse. It is suggested that keys can be checked out through the department secretary from 4:00-5:00pm only and must be returned the following work day by 9:00am.

The faculty/graduate student microcomputer room should house at least three microcomputers, three disk drives, three video screens, and three dot matrix printers. At least two of these microcomputers should have color graphics and sound capabilities, so that one can be taken into classrooms for demonstration programs leaving a second one available for walk-in use. It is recommended that this room be kept locked with keys given to all faculty and graduate students. This room should serve as a microcomputer library storing all reference manuals, users guides, and software. A separate microcomputer and letter quality printer should be located in the office area for secretarial and faculty word processing.

There are two major restrictions that should be enforced in order for this program to run smoothly for such a large number of users. First, all computer use should be restricted to one hour per person at a time. This would encourage advance preparation on the part of all users. Second, and more important, all word processing, except that done in the department office, should be restricted to after 5:00pm. It should be emphasized that the microcomputer's primary purpose is as a computational tool and not as an expensive typewriter.

It should be the department's responsibility to provide paper and ribbon for daily computer use and disks as necessary for the CE software library. Users should provide their own disks for individual use. Concerning word processing supplies, it is suggested that ASCE set up a program each semester whereby part of a student's ASCE dues go toward the purchase of paper and ribbon. Since seniors are more likely

to use the word processing feature for resumes than are freshmen and sophomores, their dues should be higher accordingly. A similar program can be set up by the graduate students at the beginning of each semester during graduate seminar to determine the fairest way to provide paper and ribbon for theses.

The available equipment currently on hand is as follows:

- 3 TRS80 Model II microcomputers
- 3 TRS80 video screens (black & white)
- 6 TRS80 disk drives
- 3 TRS80 dot matrix printers
- 1 Daisy Wheel letter quality printer (TRS80 compatible)
- 1 Apple II microcomputer (with graphics and sound)
- 2 Apple disk drives
- 1 Apple video screen (color)
- 1 Paper Tiger dot matrix printer (Apple compatible).

As the plan stands, the department would need to purchase the following additional equipment:

- 4 Microcomputers (one with a graphics feature)
- 4 Compatible disk drives
- 4 Video screens (preferably one color)
- 2 Dot matrix printers
- 1 Letter quality printer.

Some of the extra disk drives from above may be used with the new machines depending upon the type of microcomputer purchased. Even though disks are not compatible between different brands of microcomputers, programs coded in BASIC on the Apple, for example, can be translated very easily to a TRS80 disk.

These recommendations are meant to serve as a guide to incorporating microcomputers into the Civil Engineering curriculum at K-State. For this plan to be successful, however, requires the full cooperation of the faculty and department head. There are a number of grants available to

provide equipment and salary money for faculty to develop software during the summer months. In addition, many software programs are available free of charge through computer magazines, service centers, and campus computer users groups. Many of these programs can be easily adapted to the present system or modified for special department use by senior or graduate students. As the faculty gain expertise in microcomputer usage and begin incorporating it into their classroom assignments, additional equipment purchases may be necessary. No long range recommendations have been made at this time, because with the fast growth of the microcomputer industry, the department may find it advisable to trade in part or all of their existing equipment.

REFERENCES

- Baker Van Saun, John. "Software and Documentation Requirements for Effective Instructional Use of Desktop Computers." American Society for Engineering Education Conference Proceedings, Midwest Section 17th Annual Meeting, University of Missouri, Rolla, MO, March, 1982.
- Bissey, Charles R. "So You Want To Purchase Microcomputers!" American Society for Engineering Education Conference Proceedings, Midwest Section 17th Annual Meeting, University of Missouri, Rolla, MO, March, 1982.
- Bowles, Joseph E. "Is the Computer Program Giving Correct Answers?" Civil Engineering Magazine, Vol 51, No 10, American Society of Civil Engineers, New York, NY, October, 1981, p 59-61.
- Breuning, Siegfried M. "Automated Highways: What Is the Dream and How Do We Reach It?" ed. Diebold, John. The World of the Computer, Random House, Inc, New York, NY, 1973, p 187-89.
- Brewster, Keith. "Who's Afraid of the Big, Bad Matrix?". Creative Computing, Vol 7, No 12, Morristown, NJ, December, 1981, p 168-72.
- Burck, Charles G. "The Little Railroad That Could." ed. Diebold, John. The World of the Computer, Random House, Inc, New York, NY, 1973, p 179-83.
- Butalla, Martin W. "Microcomputers: Do They Have a Place in Large Engineering Firms?" Civil Engineering Magazine, Vol 52, No 6, American Society of Civil Engineers, New York, NY, June, 1982, p 80-81.
- "Buying Computer Software." Consumer's Research, Vol 66, No 2, Consumer's Research, Inc, Washington, NJ, February, 1983, p 15-16.
- Chang, C. Alec and Eastman, Robert M. "Work Methods Design With Computer Aids." American Society for Engineering Education Conference Proceedings, Midwest Section 17th Annual Meeting, University of Missouri, Rolla, MO, March, 1982.
- Cogell, Wayne. "Robert Baum's Ethics and Engineering Curricula." American Society for Engineering Education Conference Proceedings, Midwest Section 17th Annual Meeting, University of Missouri, Rolla, MO, March, 1982.



- Dallaire, Gene. "RPI's Mighty Goal: To Help Rejuvenate American Industry." Civil Engineering Magazine, Vol 51, No 10, American Society of Civil Engineers, New York, NY, October, 1981, p 54.
- Dallaire, Gene. "The Microcomputer Explosion in Civil Engineering Firms." Civil Engineering Magazine, Vol 53, No 2, American Society of Civil Engineers, New York, NY, February, 1982, p 45-50.
- Diebold, John. The World of the Computer, Random House, Inc, New York, NY, 1973, p 3-21.
- Emanuel, Jack H. and Taylor, Charles, M. "An Interactive, Computer-Graphics Program for Truss Design." American Society for Engineering Education Conference Proceedings, Midwest Section 17th Annual Meeting, University of Missouri, Rolla, MO, March, 1982.
- Garrett, James H. "No Frills Random Access for the TRSBO." Creative Computing, Vol 7, No 12, Morristown, NJ, December, 1981, p 206.
- Garrett, Michael F. "Computer-Aided Drafting in Civil Engineering." Civil Engineering Magazine, Vol 52, No 5, American Society of Civil Engineers, New York, NY, May, 1982, p 72-76.
- Greenberger, Martin. Computers, Communications, and the Public Interest, Johns Hopkins Press, Baltimore, MD, 1971, p 120.
- Hawkes, Nigel. The Computer Revolution, E.P. Dutton & Co, Inc, NY, 1972, p 73-74.
- Hommertzheim, Donald L. "Effective Use of Microcomputers in the Engineering Probability Classroom." American Society for Engineering Education Conference Proceedings, Midwest Section 17th Annual Meeting, University of Missouri, Rolla, MO, March, 1982.
- Johnson, Kenton H. "A Primer on Microcomputers." Civil Engineering Magazine, Vol 52, No 4, American Society of Civil Engineers, New York, NY, April, 1982, p 59-60.
- Kemeny, John G. Man and the Computer, Charles Scribner's Sons, NY, 1972, p 77-79.
- Kuzmanovic, Bogdan O. and Willems, Nicholas. "Minicomputers in Structural Optimum Design." Civil Engineering Magazine, Vol 52, No 5, American Society of Civil Engineers, New York, NY, May, 1982, p 82.

- Matthews, Carole B. "The Home Computer Comes of Age." Consumer's Research, Vol 65, No 11, Consumer's Research, Inc, Washington, NJ, October, 1982, p 12.
- Poole, Lon with Borchers, Mary and Castlewitz, David M. Some Common Basic Programs: Apple II Edition, Osborne/McGraw-Hill, Berkeley, CA, 1981.
- Poole, Lon with McNiff, Martin and Cook, Steven. Apple II User's Guide, Osborne/McGraw-Hill, Berkeley, CA, 1981.
- Presley, Bruce. A Guide To Programming in Apple Soft, Lawrencevill Press, Inc, New York, NY, 1982.
- Purcupile, J.C. "New Age Design Contest - Computer Simulation." American Society for Engineering Education Conference Proceedings, Midwest Section 17th Annual Meeting, University of Missouri, Rolla, MO, March, 1982.
- Sawusch, Mark. 1001 Things To Do With Your Personal Computer, TAB Books, Blue Ridge Summit, PA, April, 1980, p 132-66.
- Schaumburg, Frank D. "Engineering - 'As If People Mattered.'" Civil Engineering Magazine, Vol 51, No 10, American Society of Civil Engineers, New York, NY, October, 1981, p 56.
- Silberman, Charles E. "Technology Is Knocking at the Schoolhouse Door." ed. Diebold, John. The World of the Computer, Random House, Inc, New York, NY, 1973, p 204-20.
- Studyvin, Bill. "Computer Activities in the Laboratory." American Society for Engineering Education Conference Proceedings, Midwest Section 17th Annual Meeting, University of Missouri, Rolla, MO, March, 1982.
- Taylor, Donald J. "Neaten Up Those Messy Tables." Creative Computing, Vol 7, No 12, Morristown, NJ, December, 1981, p 202.
- "Technical Council on Computer Practices." ASCE News, Vol 8, No 2, American Society of Civil Enigneers, New York, NY, February, 1983, p 9.
- Uebelhoer, Jane. "The Role of Engineering Ethics in the Engineering Curriculum." American Society for Engineering Education Conference Proceedings, Midwest Section 17th Annual Meeting, University of Missouri, Rolla, MO, March, 1982.

Zabinski, Michael P. Introduction to TRS-80 Level II Basic and Computer Programming, Prentice-Hall, Inc, Englewood Cliffs, NJ, 1980.

APPENDIX I:

AN INTRODUCTORY COURSE IN BASIC PROGRAMMING  
FOR CIVIL ENGINEERING MAJORS

This handout is designed to be used in an introductory course in Basic programming for the microcomputer. The written material can be applied to both Apple and TRS80 microcomputers. Where differences occur, the TRS80 command words or comments will be enclosed in < > marks.

Numerous examples have been given throughout the text to illustrate various principles. These have been designed to look like the actual printout on the video screen. Since a flashing cursor cannot be printed in this handout, the ] symbol is used to represent the computer is ready for the user to type in a command.

## TENTATIVE COURSE OUTLINE

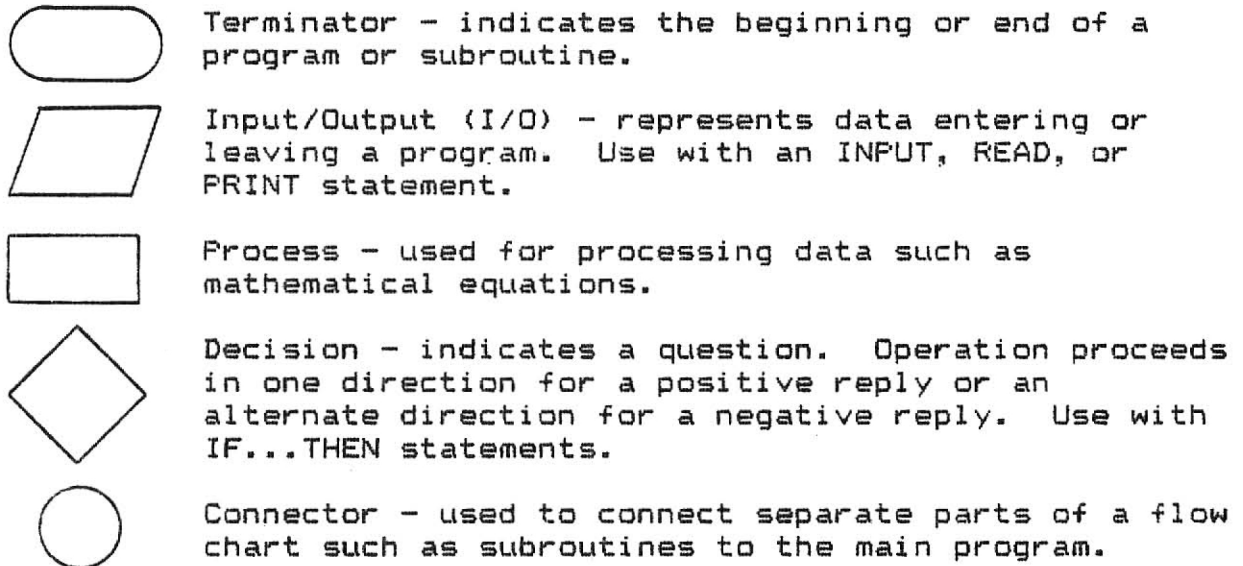
Lesson	Topic	Page
1-1	Flowcharting	38
1-2	Operating the Microcomputer	40
2-1	Introductory Program Writing in Basic	43
2-2	Decisions	48
3-1	Mathematic Functions	50
3-2	Loops	51
4-1	Single Subscripted Variables	53
4-2	Nested Loops	54
5-1	Formatting Output	55
5-2	Subroutines	58
6-1	Integers & Random Numbers	60
6-2	Quiz - Basic	
7-1	Arrays	62
7-2	Immediate Mode Programming	63
8-1	Sequential Files	64
8-2	Sequential Files (cont)	66
9-1	Random Access Files	68

**THIS BOOK  
CONTAINS  
NUMEROUS PAGES  
WITH DIAGRAMS  
THAT ARE CROOKED  
COMPARED TO THE  
REST OF THE  
INFORMATION ON  
THE PAGE.**

**THIS IS AS  
RECEIVED FROM  
CUSTOMER.**

## 1-1 FLOWCHARTING

A flowchart is an outline in symbol form. The basic symbols are:



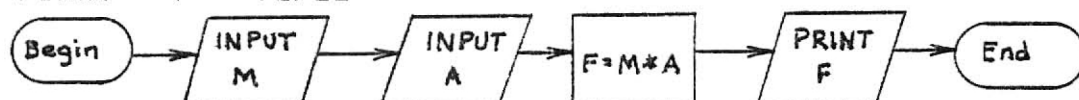
The symbols are connected by arrows to indicate the order of operation. Flow should always proceed from left to right or top to bottom.

Simple instructions are given inside each symbol. Within a process box, the words INPUT and READ are used to call data into a program and the word PRINT is used to output data from a program. The following characters are used to represent mathematical operations:

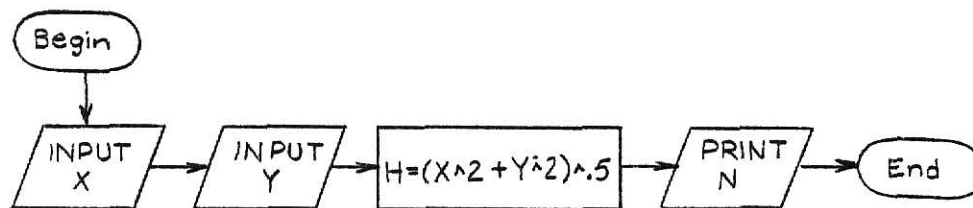
- + Addition
- Subtraction
- \* Multiplication
- / Division
- $\wedge$ N Power (to the Nth power)

Examples:

1. Given: M = mass  
A = acceleration  
Find: F = force

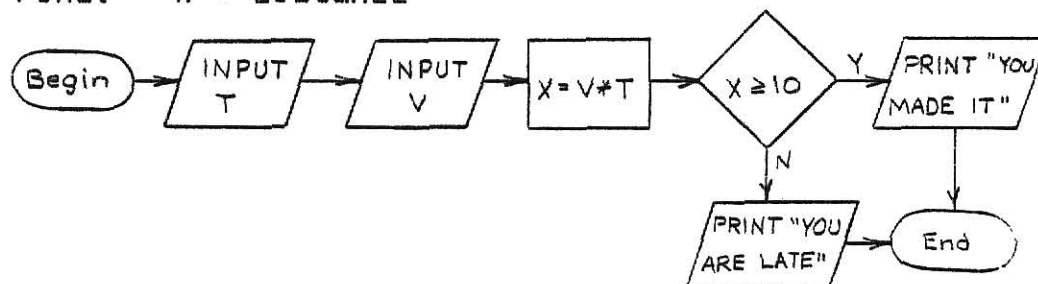


2. Given:  $90^\circ$  triangle with legs  $X$  and  $Y$   
Find:  $H$  = hypotenuse

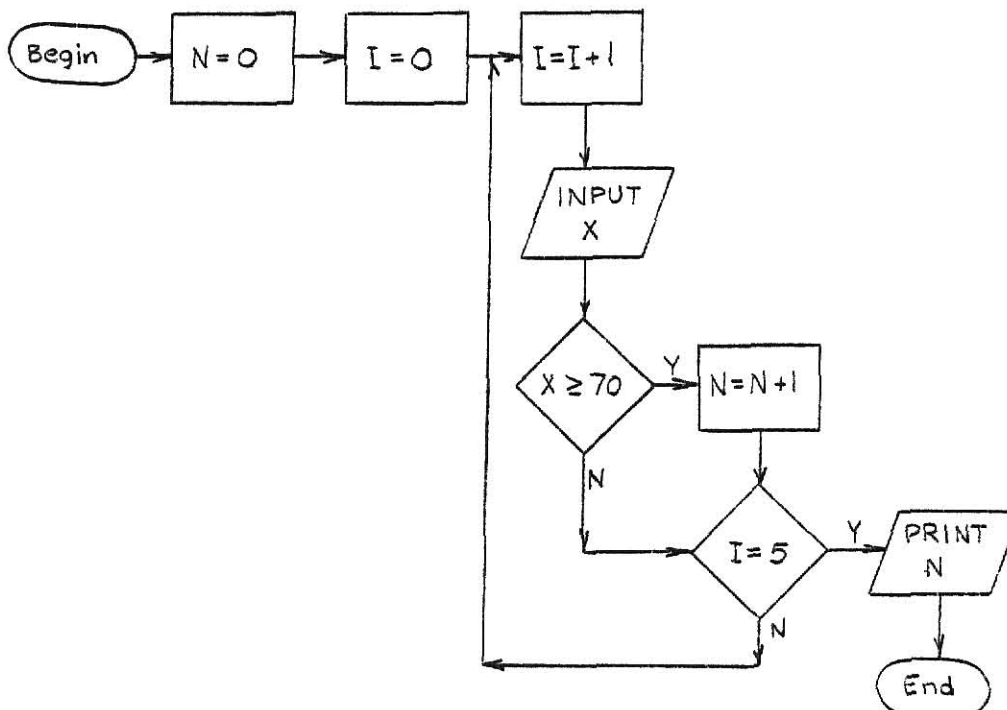


3. Given: It is 10 miles from your apartment to the campus. Can you get to school in time for your first class if you have only  $T$  (=time) minutes to get there?  
 $V$  = velocity (miles/minute)

Find:  $X$  = distance



4. Given:  $X$  = final test scores for 5 students  
 $I$  = a counter  
Find:  $N$  = number of students who passed with a score of 70 or better





## 1-2 OPERATING THE MICROCOMPUTER

Microcomputer programs are stored on diskettes. The Apple uses 5" diskettes (the TRS80 uses 8" diskettes). All diskettes are affected by strong magnetic fields (ie. TV sets), heat, or bending so care must be taken when using them.

## APPLE OPERATION

To use the Apple microcomputer, follow these instructions:

- 1) Flip up the cover of the disk drive.
- 2) Carefully slide in a diskette with the label facing up (make sure the diskette is all the way in).
- 3) Close the cover of the disk drive.
- 4) Turn on the computer. The switch is in the back on the left side.
- 5) Turn on the video screen. The switch is in the front.
- 6) Turn on the printer if it is to be used. The switch is in the back on the left side. Be sure the switch on the top right side is pointed toward ON-LINE and the red light is on.
- 7) To enter a new program, type NEW, press RETURN, and begin typing your program code. To run an existing program, type RUN programname and press RETURN.

If the computer is turned on before a diskette has been inserted, loud clacking sounds will be heard. Turn the computer off and insert a diskette. Never insert an uninitialized diskette.

When using a diskette for the very first time, it must be initialized.

- 1) Insert the System Master diskette.
- 2) Turn on the computer and video screen.
- 3) When the blinking cursor appears on the screen, remove the System Master diskette from the disk drive and replace it with a blank diskette.
- 4) Type NEW and press RETURN.
- 5) Now type a greeting program. Any program can be used. The following is a very simple one to start with. Press RETURN after each line and supply your own name and date.

```

10  REM HELLO
20  PRINT "DISKETTE CREATED ON 64K SYSTEM"
30  PRINT "BY JOHN DOE ON 1/12/83"
40  PRINT CHR$(4); "CATALOG"
50  END

```

- 6) Type INIT HELLO. Press RETURN. The diskette will spin for nearly a minute.
- 7) When the diskette stops spinning and the IN USE light goes off on the disk drive, remove your diskette and label it (write gently!).

A diskette needs to be initialized only once. It is now ready for use.

#### <TRS80 OPERATION

To use the TRS80 microcomputer, follow these instructions:

- 1) Turn on the master switch located on the extension cord outlet box.
- 2) When the message INSERT DISKETTE appears on the video screen, open the disk drive door.
- 3) Carefully slide in a diskette with the label facing to the right (make sure the diskette goes all the way in).
- 4) Slam the disk drive door. Whirring noises will start.
- 5) Answer date and time questions.
- 6) The message TRSDOS READY will appear. Type in the word BASIC and press ENTER.
- 7) To enter a new program, type NEW, press ENTER, type AUTO (for automatic line numbering), press ENTER, and begin typing your program code.
- 8) To run a program in memory, type LOAD, the program name, and press ENTER.
- 9) Type RUN and press ENTER.

When using a diskette for the very first time, it must be formatted.

- 1) Follow steps 1 thru 5 from the TRS80 operating instructions using the Basic Master diskette.
- 2) When the TRSDOS READY message appears, type in FORMAT and press ENTER.
- 3) Insert a new diskette in disk drive 1.
- 4) Answer the drive location questions. The formatting takes about ten minutes, so be patient.
- 5) When the formatting is completed, type in BACKUP and press ENTER.
- 6) Answer the drive location, name, and password questions. Note, that the password is optional.
- 7) When the backup is completed, remove the new disk from drive 2 and label it.

A diskette needs to be initialized only once. It is now ready for use.>

## MICROCOMPUTER COMMANDS

The following is a list of commands to use when manipulating a program from a diskette.

RETURN <ENTER> Button - This button is pressed after typing in a command to enter data into memory. It is pressed after each line of a program.

NEW - This should be typed before typing in a new program. It clears the memory of all old data. It should not be typed, however, when editing an existing program or the program will be erased.

SAVE - After typing in a program, this command should be typed followed by a program name. The program name can be any name consisting of letters, numbers, and/or more than one word, but each program must have a different name.

LOAD - This command loads an existing program into memory. The program name must follow the word LOAD.

RUN - This command executes a program in memory.

CATALOG - This command gives a listing of all the programs in memory.

LIST - This command is used to print out (on the video screen) the lines of a program in memory.

LIST	Entire program is displayed.
LIST 20	Line 20 is displayed.
LIST 20-80	Lines 20 through 80 inclusive are displayed.

## 2-1 INTRODUCTORY PROGRAM WRITING IN BASIC

A program is a set of instructions, called commands, for the computer. These commands must be written in a logical order using words and symbols the computer can understand.

## LINE NUMBERS

Commands are grouped together to form statements or lines of a program. Each statement can contain a maximum of 239 <255> characters.

Each command statement must be numbered. The computer reads the line numbers in ascending order, from lowest to highest. The largest line number allowed is 63999 <65529>. Statements can be entered in any numerical order. The computer will automatically sort them before executing the program. It is a good idea initially to number statements in multiples of 10 so that additional commands can be inserted easily at a later date.

```
ie.  10  INPUT A
      20  PRINT N
      30  N = 4*A
```

Note that this program contains a "logic" error. Line 20 asks the computer to print the value of N, but that value is not calculated until line 30. Therefore, the computer will print a zero for N.

<The TRS80 has an automatic line numbering feature. Typing AUTO before entering the program will cause each statement to be incremented by 10. To stop the line numbering feature, press BREAK. Variations of this feature are shown below:

```
AUTO          Begin line #10; increment = 10.
              10, 20, 30...

AUTO 50        Begin line #50; increment = 10.
              50, 60, 70...

AUTO 50,5      Begin line #50; increment = 5.
              50, 55, 60...>
```

## VARIABLES

There are three types of variables in Basic: real, integer, and string. Variable names may contain any letters or numbers, but the computer reads only the first two as the variable name. The first character must be a letter.

Variable names cannot contain any reserved words. Integer variable names are distinguished by a percent sign, %, anywhere after the first character. String variable names contain a dollar sign, \$, anywhere after the first character. Real variable names do not require special characters.

```
ie.  D1 = 6.321      Real
      NU = 24.1E6     Real
      NUMBER = 24.1E6 Real
      X% = 74         Integer
      NUMBER% = 327   Integer
      A$ = "COMPUTER" String
      NAME$ = "JOHN"  String
```

Note that NU and NUMBER represent the same variable name, NU, to the computer; NUMBER%, however, represents a different variable, NU%, due to the % sign.

Real variable values can be entered in decimal, integer, or exponential form.

```
ie.  X = 1.141      Decimal
      Y = 20         Integer
      Z = 1.345E3    Exponential
```

Integer variables do not contain decimals. If decimal values are entered for an integer variable, the decimal portion is truncated to the next lower whole number.

```
ie.  I% = 1.141      I% = 1
      LA% = 5.967     LA% = 5
      NU% = -6.312    NU% = -7
```

Integer values cannot be less than -32767 <-32768> nor greater than +32767. Both integer and real variables are initially equal to zero until they are specified otherwise.

String variables may contain any combination of letters, numbers, and/or symbols up to a maximum of 239 <255> characters. All characters must be enclosed in quotes (the quotes are not printed).

The user must be careful to distinguish between a zero and a capital letter "O"; zeros are often characterized by a slash through the number.

## OPERANDS

There are six basic symbols used to perform mathematical operations. The computer recognizes a natural order of operation when analyzing equations:

- 1) ^ Power functions anywhere within an equation are performed first.

- 2) \*    Multiplication and division are performed equally after power functions in the order they occur when reading from left to right.
- 3) +    Addition and subtraction are performed equally after powers, multiplication, and division in the order they occur from left to right.
- 4) ( )    Parenthesis are used to change the natural order of operation. Any function enclosed within parenthesis will be performed before all others.
- 5) =    Remember when calculating an equation, the variable name must be to the left of the equal sign and the mathematic expression to the right.  
ie.  $R = 2*3$         NOT     $2*3 = R$

Examples:

$\frac{2AB+C}{4DF}$             is the same as  $(2*A*B+C)/4/D/F$

$\frac{5A^2 + \frac{1}{2}}{8-B}$             is the same as  $(5*A^2+(1/2))/(8-B)$

#### COMMAND WORDS

The PRINT command can print both numbers and words. The word PRINT is followed by an "argument". If an argument is enclosed in quotes, the message within the quotes will be printed as is. Arguments without quotes will print the value they stand for. A question mark can be used as an abbreviation for the PRINT command. When the program is listed, the computer will replace the question mark with the command word PRINT.

```
ie. 10  PRINT 3*2
     20  PRINT "3*2"
     30  R = 4
     40  ? R
```

```
IRUN
6
3*2
4
```

The INPUT command calls data into the machine through the keyboard. When the computer reaches an INPUT statement, execution halts until the user responds by typing in an

answer. One phrase can be printed out with the INPUT command in order to let the user know when data has been requested.

```
ie.  50  INPUT "DIAMETER="; D
      60  INPUT "PI="; PI
      70  A = PI*D^2/4
      80  PRINT "CIRCLE AREA = ";A
```

```
IRUN
DIAMETER=74
PI=73.141
CIRCLE AREA = 12.5641812
```

Note the question marks after DIAMETER= and PI=. The computer automatically prints a question mark with a flashing cursor to let the user know that a response is required. This should be distinguished from the I symbol in that the question mark is printed during program mode while the I symbol is printed during immediate mode.

The user can input multiple data with one command by separating the variables and their values by commas.

```
ie.  10  INPUT "ENTER TEST SCORES: ";A,B,C,D
      20  AVG = (A+B+C+D)/4
      30  PRINT "AVERAGE = "; AVG
```

```
IRUN
ENTER TEST SCORES:  ?100,85,78,96
AVERAGE = 89.7512142
```

String values can be entered from an INPUT statement. However, if the string contains a comma, leading blank, or colon, it must be preceded by quotes (ending quotes are not required). The quotes will not be included in the value.

```
ie.  40  INPUT "NAME: "; N$
```

```
IRUN
NAME:  ?"SMITH, FRED R.
```

The REM (short for remark) command is used for the programmer to make explanations to clarify a program. Everything entered after the REM is ignored by the computer. The REM statement is not printed during program execution; it only appears when the program is listed. It is a good practice for the programmer to use two REM statements at the beginning of each program to indicate the program name and the programmer.

```
ie.  10  REM  PROGRAM NAME:  AREA OF A CIRCLE
      20  REM  PROGRAMMER:  JOHN DOE
```

<An apostrophe can be used as an abbreviation from the REM command on the TRS80. When the program is listed, the

computer will replace the apostrophe with the command word REM.>

The END command is the last statement in the main program telling the computer the program is finished.

```
ie. 80 PRINT 2*3
     90 END
```

```
IRUN
6
```

#### COMMAND PUNCTUATION

Commas are used to separate a series of variables in INPUT statements.

```
ie. 10 INPUT X,Y,Z
```

Semicolons separate phrases in quotes from variable names in PRINT statements. The value of the variable is displayed directly after the last character in quotes, therefore the user may need to leave a blank character for readability.

```
ie. 50 PRINT "THE FORCE ="; P
     60 PRINT "THE AREA EQUALS "; PI*D^2/4
     70 PRINT "THE STRESS EQUALS "; P/A
```

```
IRUN
THE FORCE =100
THE AREA EQUALS 5
THE STRESS EQUALS 20
```

<Note: The TRS80 automatically prints one blank character before every positive number. This blank is filled by a hyphen for negative numbers.

```
ie. 50 PRINT "THE FORCE EQUALS"; P
     60 PRINT "THE FORCE EQUALS"; -P
```

```
IRUN
THE FORCE EQUALS 100
THE FORCE EQUALS-100>
```

Colons are used to separate multiple commands within the same line number.

```
ie. 10 INPUT X: PRINT X
```



## 2-2 DECISIONS

The PRINT and INPUT statements are unconditional; that is, they are processed once in the order they occur. There are times when the user may want to repeat an operation. At other times, the user may want to perform one operation if the answer to a question is "yes" and another operation if the answer to the question is "no".

The GOTO command is used as a form of branching to skip or repeat steps. It is followed by a line number to indicate where operation is to resume.

```
ie. Find the stress for multiple loads.
10  A = 5
20  INPUT "LOAD = "; P
30  PRINT "STRESS = "; P/A
40  GOTO 20
```

An IF...THEN command is used to ask a question. A "condition" follows the word IF and a line number follows the word THEN. If the condition is true, program execution will branch or GOTO the line number indicated (the GOTO is implied). If the condition is false, operation proceeds to the line number immediately following the IF...THEN statement.

A condition consists of two quantities separated by one of the following symbols:

```
=    Equal to
>    Greater than
<    Less than
>=   Greater than or equal to
<=   Less than or equal to
<>   Not equal to
```

(Note that =, >, <, or <> can be used just as effectively.)

The quantities can be variables, numbers, alphanumeric characters enclosed within quotes, or equations.

```
ie. Enter two numbers and print the larger.
```

```
10  INPUT "ENTER 2 NUMBERS "; X1,X2
20  IF X1>X2 THEN 50
30  PRINT X2; " IS THE LARGER"
40  GOTO 60
50  PRINT X1; " IS THE LARGER"
60  END
```

To be more flexible, the user can immediately execute a command with a true condition. Notice how two statements have been eliminated from the previous program.

```
ie.  10  INPUT "ENTER 2 NUMBERS "; X1,X2
      20  IF X1>X2 THEN PRINT X1; " IS THE
          LARGER": GOTO 40
      30  PRINT X2; " IS THE LARGER"
      40  END
```

Remember, any commands following the IF...THEN statement on the same line number will be ignored if the condition is false.

The user may wish to have multiple conditions within one IF...THEN statement. Multiple conditions are separated by the word AND or the word OR. The AND means all conditions must be true in order to execute the subsequent operation. The OR means at least one condition must be true.

ie. Enter a number and determine if it is between 21 and 52.

```
10  INPUT "ENTER A NUMBER "; X
20  IF X>21 AND X<52 THEN PRINT X; " IS BETWEEN
    21 AND 52": GOTO 40
30  PRINT X; " IS OUT OF RANGE"
40  END
```

Conversely, enter a number and determine if it does not fall between 21 and 52.

```
10  INPUT "ENTER A NUMBER "; X
20  IF X<=21 OR X>=52 THEN PRINT X; " IS OUT OF
    RANGE": GOTO 40
30  PRINT X; " IS BETWEEN 21 AND 52"
40  END
```

## 3-1 MATHEMATIC FUNCTIONS

The sine, cosine, or tangent of an angle can be found using the commands SIN(X), COS(X), or TAN(X) where X stands for any angle measured in radians. To convert an angle in decimals to radians, use the formula:

$X(\text{radians}) = X(\text{decimals}) * 3.14159265/180$   
 or  $X(\text{radians}) = X(\text{decimals}) * 0.0174533$

- ie. Resolve the force, F, at an angle, A, measured from the X axis into its X and Y components.

```
10 INPUT "ENTER FORCE, ANGLE (IN DECIMALS)"; F,A
20 A = A*3.141/180
30 PRINT "F(X)="; F * COS(A)
40 PRINT "F(Y)="; F * SIN(A)
```

The only inverse trigonometric function is the arctangent or inverse tangent. To find the angle, in radians, whose tangent is (X), type ATN(X). To convert radians to decimals, multiply by 57.29578.

To find the square root of a number, N, type SQR(N). This expression is identical to  $N^{.5}$ , but the computer execution is faster. N can stand for any arithmetic expression as long as the result is a non-negative number.

- ie. Find the hypotenuse of a right triangle of base, X, and side, Y.

```
10 INPUT "BASE, SIDE "; X,Y
20 PRINT "HYPOTENUSE="; SQR(X^2 + Y^2)
```

To return the absolute value of a number, N, type ABS(N). This function can be used with the SQR function to insure a positive argument.

- ie. 10 PRINT SQR(ABS(7 - 16))

```
JRUN
3
```

To find the natural log of a number, N, (log to the base e), type LOG(N). N must be a positive number. To find the common logarithm (log to the base 10), type LOG(N)/LOG(10). The 10 can be replaced with any base number for alternate logarithms.

To use the exponential function, e to the Xth power (where e = 2.71828), type EXP(X).

## 3-2 LOOPS

## FOR/NEXT STATEMENTS

Another way of repeating a sequence of steps is by using a FOR/NEXT loop. The format is:

```
FOR variable = start value TO end value STEP increment
.
. (statements within the loop)
.
NEXT variable
```

The variable serves as a counter and can be used within the program. The start value is usually zero or one, but any value can be used. The STEP increment is the value that is added to the counter after one run through the loop. The STEP increment can be omitted if the increment equals +1.

```
ie. 50 FOR I=0 TO 3
      60 PRINT I
      70 NEXT I
```

```
JRUN
0
1
2
3
```

```
ie. 80. FOR J = 1 TO 5 STEP 2
      90 PRINT J: NEXT J
```

```
JRUN
1
3
5
```

To run through a series of numbers backwards, the start value must be the larger number and the end value must be the smaller. The increment must be a negative number.

```
ie. FOR I=5 TO 1 STEP -1      I = 5,4,3,2,1
      FOR J=5 TO 3            J = 5 only
      FOR P=1 TO 8 STEP -1    P = 1 only
      FOR E=6.1 TO 6.3 STEP .1 E = 6.1,6.2,6.3
```

## READ/DATA STATEMENTS

There are times when the user may wish to store data inside the program rather than keying it in each time the program is run. In this case, the READ/DATA commands replace

the INPUT command.

```
ie.  10  READ X, Y, Z
      20  DATA 10,7,30
```

Values are assigned to variable names in the order they are listed: X=10, Y=7, Z=30. DATA statements can be located anywhere within a program. When all the values from one DATA statement have been assigned, the computer advances to the next DATA statement. Subsequent READ statements begin assigning variables where the last statement left off.

```
ie.  10  READ X, Y
      20  DATA 20,40,60,80,100
      30  READ A, B
```

```
X=20
Y=40
A=60
B=80
```

DATA statements can store numbers or alphanumeric characters. Alphanumeric phrases should be enclosed in quotes if they contain a comma, leading blank, or colon. The quotes are not included in the value. DATA statements will not compute arithmetic operations such as 3\*2.

```
ie.  10  READ A$,BN
      20  DATA HARRY,8.5E6
```

```
A$=HARRY
BN=8.5E6
```

The RESTORE command causes the READ statement to begin reading at the beginning of the DATA statement. This is especially desirable when searching for one piece of data.

```
ie.  10  INPUT "NAME ";A$
      20  RESTORE
      30  FOR I=1 TO 4
      40  READ B$, SCORE
      50  IF B$=A$ THEN PRINT B$;"-";SCORE
      60  NEXT I
      70  DATA BARNES,100,JONES,89,MILLER,95,
           SMITH,83
      80  GOTO 10
```

```
IRUN
NAME ?MILLER
MILLER-95
```

```
IRUN
NAME ?BARNES
BARNES-100
```

## 4-1 SINGLE SUBSCRIPTED VARIABLES

There are times when the user will need to manipulate multiple values of one property such as a variable diameter rod. Rather than assigning individual variable names to each value, the programmer can dimension a single subscripted variable. The DIM A(X) command is used to reserve (X+1) places in memory for the variable A (this includes the zeroth place).

```
ie.  10  DIM SC(4)
      20  FOR I=1 TO 4
      30  INPUT "ENTER A TEST SCORE";SC(I)
      40  SC(0) = SC(0)+SC(I)
      50  NEXT I
      60  PRINT "THE AVERAGE = ";SC(0)/4
```

In the above program, four places are reserved for the test scores and the zeroth place for the average. Dimensioning a variable automatically sets the initial values equal to zero. It should be noticed that A(1) and A1 are completely different variables and can be used within the same program to represent different properties.

It is possible to have several DIM statements within one program, but each variable can be dimensioned only once. An attempt to dimension the same variable twice will result in an error message. Multiple DIM variables on the same line are separated by commas.

```
ie.  50  DIM A$(5),BAL(10),V(7)
      60  DIM X(2)
```

The depth of a DIM variable can be specified by either a number or a numerical expression provided any variables within the expression have been specified beforehand.

```
ie.  10  INPUT "NUMBER OF TEST SCORES=";N
      20  DIM SC(N),A(N+2)
```

## 4-2 NESTED LOOPS

FOR/NEXT loops can be used in groups of two or more to perform multiple operations. This technique is especially useful when a problem is solved by trial and error.

```
ie.  30  FOR I=1 TO 2
      40  FOR J=2 TO 10 STEP 2
      50  PRINT J*I;" ";
      60  NEXT J
      70  NEXT I
```

```
IRUN
  2 4 6 8 10 4 8 12 16 20
```

Nested loops can never cross. The inner loop must be entirely contained within the outer loops. To check for loops crossing, draw brackets outlining the range of each loop. If the brackets do not cross, the loops do not cross.

```
ie.   90  N = 40
      100  FOR L1=1 TO 10
      :
      150  FOR L2=1 TO 19 STEP 2
      :
      200  NEXT L2
      210  FOR L3=5 TO N STEP 5
      :
      400  FOR L4=10 TO 1 STEP -1
      :
      480  NEXT L4
      490  NEXT L3
      :
      500  NEXT L1
```

Multiple NEXT commands can be combined on one line by separating the variables by commas. Be sure the variables are listed in the proper order (inner loop to outer loop).

```
ie.  30  FOR I=1 TO 5
      40  FOR K=10 TO 20
      :
      90  NEXT K,I
```

## 5-1 FORMATTING OUTPUT

## PUNCTUATION

There are three natural column zones on the video screen <four zones on the TRS80> and five column zones on the printer. When variables in PRINT statements are separated by commas rather than semicolons, the output will tab to the next column zone.

```
ie.  50  PRINT "NAME", "GRADES", "AVERAGE"
      60  PRINT "JONES", 100, 100
```

```
IRUN
NAME          GRADES          AVERAGE
JONES         100             100
```

Note that if a variable output completely fills a column zone or is within three characters of the next zone, the output will skip to the third zone.

When variables are separated by semicolons, the first character of the second variable is printed in the space directly after the first variable. <Remember, the TRS80 automatically prints one blank character before every positive number. This blank is filled by a hyphen for negative numbers.> A "hanging semicolon" can be useful when printing the output from two different statements on one line.

```
ie.  10  PRINT "THE AREA = ";
      20  AREA = 3.141*12^2/4
      30  PRINT AREA
```

```
IRUN
THE AREA = 113.076
```

## APPLE TAB FUNCTIONS

The video screen is 40 columns wide by 24 lines deep. The TAB(X) command is similar to a tab button on a typewriter. It allows output to be printed at specified locations.

The TAB(X) command must be used within a PRINT statement. The left edge of the screen is TAB(1); the right edge is TAB(40).

```
ie.  10  PRINT TAB(10);"NAME";TAB(20);492-54-8071
```

```
IRUN
NAME          492-54-8071
```

It is important to separate the TAB(X) command and the variable or comment in quotes by semicolons.



The HTAB(X) and VTAB(X) commands are used without PRINT statements. They are used to move the cursor around the screen. HTAB(1) moves the cursor to the left of the screen; HTAB(40) moves the cursor to the right. Unlike the Print TAB(X) command which can only move the cursor progressively to the right on any one line, the HTAB(X) command can move the cursor to the right or left. Similarly, VTAB(1) moves the cursor to the top of the screen; VTAB(24) moves the cursor to the bottom.

#### <TRS80 TAB FUNCTIONS

The video screen is 80 columns wide by 24 lines deep. The TAB(X) command is similar to a tab button on a typewriter. It allows output to be printed at specified locations.

The TAB(X) command must be used within a PRINT statement. The X variable stands for any value from 0 to 255. The left edge of the screen is TAB(0). Values of X greater than 79 are printed on subsequent lines.

```
ie. 10 PRINT "NAME";TAB(20) "GRADE";TAB(80)
      "JONES";TAB(100) 100
```

```
IRUN
NAME          GRADES
JONES         100
```

Notice that no space is allowed between the TAB and the left parenthesis. No punctuation is required after the TAB(X) command.

The Print @ X, command is used to move the cursor around the screen. X stands for any number between 0 and 1919 (1919 corresponds to the number of possible cursor positions on the screen: 80 columns \* 24 lines = 1919). The argument follows the comma.

```
ie. 10 PRINT @ 30,6
      (prints a 6 on line 1, col 31)
    20 Print @ 0,X
      (prints the value of X on line 1, col 1)
    30 PRINT @ 272,"LOADS"
      (prints the word LOADS on line 4, col 32)>
```

#### MENUS

A menu is a computer term to represent a group of options.

```
ie.          MENU

      (1) FIND X-FORCES
```

## (2) FIND Y-FORCES

## SELECT ONE:

Program execution should branch to the appropriate routine depending upon the user's selection.

Before constructing a menu, the programmer should clear the screen of all other output. The HOME <CLS> command erases the screen and places the cursor in the upper left-hand corner.

```
ie.  10  HOME
      <10  CLS>
      20  PRINT TAB(18);"MENU"
      <20  PRINT TAB(18)"MENU">
      30  PRINT "(1)  FIND X-FORCES"
      40  PRINT "(2)  FIND Y-FORCES"
      50  PRINT "SELECT ONE:"
      60  INPUT N
```

The GET <INKEY\$> command can be used in place of the INPUT command on line 60 above. When the GET <INKEY\$> command is used, the computer will accept only one keystroke. The user does not have to press RETURN <ENTER> and the character does not print on the screen. Once a key has been pressed, control passes immediately to the next command. <The INKEY\$ must be placed within a loop. It will not wait for the user to press a key, but will immediately assume INKEY\$ = a null character (blank)>.

```
ie.  60  GET N
      <60  N$=INKEY$
      65  IF N$=" " THEN 60>
```

The GET <INKEY\$> command is very useful when a pause is required to let the user read a set of instructions.

```
ie.  100  PRINT "PRESS THE SPACE BAR TO CONTINUE"
      110  GET X$
      <110  X$=INKEY$
      115  IF X$=" " THEN 110>
```

## 5-2 SUBROUTINES

At times, a user may need to repeat an operation at different locations within a program. The statements comprising this operation can be isolated into a subroutine and accessed from the main program at any time.

A subroutine is accessed by the command GOSUB N, where N stands for the beginning line number of the subroutine. N should be greater than the END statement line number. A subroutine is exited by the RETURN command. This command, which is the last statement in a subroutine, returns program execution back to the line number immediately following the GOSUB command.

```
ie.  10  INPUT "ENTER AN ANGLE IN DECIMALS: "; AD
      20  GOSUB 100
      30  PRINT "THE ANGLE IN RADIANS = "; AR
      40  END

      100  AR=AD*3.141/180
      110  RETURN
```

The difference between a GOSUB and a GOTO is that a GOSUB will return to the statement following the GOSUB command. A GOTO will not.

There are other advantages to using subroutines. It is much easier to write several short subroutines than one long program. Subroutines can be tested and "debugged" separately which helps to locate errors quickly. Modifications to a program are simpler since subroutines can be added, changed, or deleted more easily than the entire program can be altered.

## ON...GOTO/ON...GOSUB

A GOTO or GOSUB branches to one specified line number; but an ON...GOTO or ON...GOSUB goes to one of several line numbers depending upon the value of a specified variable.

```
ie.  ON X GOTO N1,N2,N3
```

(where X stands for a variable and N1, N2,  
and N3 stand for line numbers)

If the value of the variable is one, execution branches to the first line number; if the value is two, execution branches to the second line number; if the value is three, execution branches to the third line number; etc. These two commands are very useful when developing menus and save the programmer from using a lot of IF...THEN...GOTO statements.

```
ie.  10  PRINT "(1) FIND ANGLE IN RADIANS"
```

```
20 PRINT "(2) FIND ANGLE IN DEGREES"  
30 PRINT "SELECT ONE:"  
40 INPUT A  
50 ON A GOSUB 100,200
```

If the user selects option 1 in the above program, execution branches to the subroutine beginning on line 100. If the user selects option 2, execution branches to line 200. Note that the ON...GOTO or ON...GOSUB command cannot be used with string variables.

## 6-1 INTEGERS &amp; RANDOM NUMBERS

## INT

The INT(X) command changes a real number, X, into an integer by truncating the decimals. X can be a number or a variable.

```
ie. 10 A=3.141
    20 PRINT INT(A)
```

```
IRUN
3
```

Note that the INT command does not round off a decimal, but truncates it leaving the smaller whole number.

```
ie. INT(4.99) = 4
    INT(10.5) = 10
    INT(-3.1) = -4
```

To round a number to the next larger whole number, if the decimal portion is greater than or equal to 0.5, use the formula  $\text{INT}(X + .5)$ .

```
ie. INT(3.4 + .5) = INT(3.9) = 3
    INT(3.5 + .5) = INT(4.0) = 4
    INT(3.6 + .5) = INT(4.1) = 4
```

In engineering calculations, the user will often want to round a number to 2 or 3 decimal places. A general formula for rounding a variable, X, to N decimal places is  $\text{INT}(10^N * X + .5) / 10^N$ .

```
ie. INT(10*2.57+.5)/10 = 2.6
    INT(10^2*38.716237+.5)/10^2 = 38.72
```

## APPLE RND

The RND(X) command will generate a positive random number between 0 and 1. X can be any positive number, including zero. (The value of X does not affect the numbers that are generated as long as X is positive.)

```
ie. R=RND(1)      R=.776733737
    S=RND(0)      S=.479674134
```

Combining the RND function with the INT function allows the user to generate random numbers greater than one.

```
ie. INT(10*RND(1)) = 3
    INT(100*RND(0)) = 21
```

Be sure both the RND argument and the INT argument are enclosed in parenthesis. To ensure a random sequence when using RND(X), X should always be a positive number. A negative value for X will generate a repeating sequence of numbers.

#### <TRS80 RND

The RND(X) command will generate a positive random number between zero and X. X can be any number greater than or equal to zero and less than 32768.

```
ie.  R=RND(1)           R=0.789465435
      S=RND(5)           S=3.79834234
      T=RND(1000)        T=389.398751
```

Combining the RND function with the INT function allows the user to generate whole numbers or significant figures.

```
ie.  INT(RND(5)) = 3
      INT(10*RND(5)) = 3.7
```

Be sure both the RND argument and the INT argument are enclosed in parenthesis. The command word RANDOM should be typed early in the program to ensure generating a non-repeating sequence of numbers.

```
ie.  10  RANDOM
      20  PRINT RND(1)
```

```
IRUN
      .261322896>
```

## 7-1 ARRAYS

Sometimes it is useful to specify variables in more than one dimension. When constructing a table, the user may want to use two dimensions. The command DIM X(A,B) reserves space in memory for the variable X with A+1 rows and B+1 columns.

```
ie.  DIM X(2,3) =
      X(0,0)      X(0,1)      X(0,2)      X(0,3)
      X(1,0)      X(1,1)      X(1,2)      X(1,3)
      X(2,0)      X(2,1)      X(2,2)      X(2,3)
```

Variables can be dimensioned all the way to 88 dimensions.

```
ie.  DIM L(3,6,2,4,7,5)
```

The more dimensions, however, the smaller each dimension size must be since memory space is limited inside the computer.

The FRE(0) command will return the number of free "bytes" left in memory available to the user. (A byte is a memory space). If there are more than 32767 bytes available, the FRE command will return a negative number. To get a positive answer, add 65536 to the negative response.

```
ie.  IPRINT FRE(0)
      -29285

      IPRINT FRE(0)+65536
      36251
```

This command can be used within programs having multiple arrays to keep track of the memory being used.

## 7-2 IMMEDIATE MODE PROGRAMMING

Most commands can be performed immediately without writing a program. The exceptions are INPUT, READ, and DATA. When a command is typed without a line number, it is executed immediately by the computer. There is no need to type in the word RUN.

```
ie.  IPRINT 3*5+5
      20
```

When working a problem in immediate mode, the user should type NEW to clear the memory of all other data. All equations should be set equal to a variable for recall at a later date. Once a value has been assigned to a variable, the computer will save that value until the memory is recleared or the computer turned off.

```
ie.  IDIA=5
      IPI=3.141
      IAREA=PI*DIA^2/4
      IS=100/AREA
      IPRINT S
      19.63125
```



## B-1 SEQUENTIAL FILES

Setting up a file enables the user to save information on a disk for use at a later date. A sequential file stores data line by line and retrieves it in exactly the same order. This type of file is somewhat time consuming for large quantities of data, since the entire file must be "dumped" into the computer memory each time it is used. Therefore, it is a good practice to limit the size of any one sequential file.

## DOS COMMANDS

All commands referring to operation of the disk drive are Disk Operating System (DOS) commands.

ie. JCATALOG

In program mode, a DOS command can be executed if it is placed within a PRINT statement. The PRINT must be followed by a CTRL-D and the DOS command word enclosed in quotes. The user could physically press the CTRL button and the D button at the same time, but they would not appear when the program was listed. A better way is to use an ASCII (pronounced "as-key-two) character conversion. All characters and numbers on the keyboard are assigned an ASCII code number. The code number for CTRL-D is 4.

The command CHR\$(N) will convert the ASCII code number, N, in parenthesis to its character equivalent. Notice the difference in the two examples:

```
ie. 10 PRINT "CATALOG": REM NO CTRL-D
      JRUN
      CATALOG
```

```
ie. 10 PRINT CHR$(4); "CATALOG"

      JRUN
      A 006 BOOT3
      B 006 APPLESOFT
      A 002 TEXT
```

If the CHR\$(N) command is to be used many times within a program, the user may wish to set it equal to a string variable for ease in keying.

```
ie. 10 D$=CHR$(4)
      20 PRINT D$; "CATALOG"
```

## FILE COMMANDS

OPEN instructs the computer to open a file and ready it

for use. A file name must follow the word OPEN.

```
ie. 10 PRINT CHR$(4); "OPEN STEEL"
```

This command tells the computer to look for a file named STEEL. If there is no file in memory under this name, the computer will create a new file. Any name can be used as a file name; however, file names must be different from program names since they are stored on the same disk.

All files must be closed after they have been opened to insure proper retention of all data.

```
ie. 20 PRINT CHR$(4); "CLOSE STEEL"
```

WRITE is used to store data in the file. All PRINT statements listed after the WRITE command send information to the file instead of the screen or printer until the file is closed or another DOS command is executed. Note that the PRINT statements do not require ASCII character conversions.

```
ie. 10 PRINT CHR$(4); "OPEN STEEL"
    20 PRINT CHR$(4); "WRITE STEEL"
    30 FOR I=1 TO 3
    40 PRINT N$(I); X$(I)
    50 NEXT I
    60 PRINT CHR$(4); "CLOSE STEEL"
```

READ is used to take data out of the file and put it in memory. All INPUT statements after the READ command take information from the file rather than the keyboard. It is important that the user remember the order the data is placed in the file as it must be accessed in the same order.

```
ie. 70 PRINT CHR$(4); "OPEN STEEL"
    80 PRINT CHR$(4); "READ STEEL"
    90 FOR I=1 TO 3
   100 INPUT N$(I), X$(I)
   110 NEXT I
   120 PRINT CHR$(4); "CLOSE STEEL"
```

The previous program could store the following information:

```
CHANNEL SECTION
C12X34
W SHAPE
W14X112
ANGLE
L12X30
```

Remember, that the user cannot access any one piece of information directly from a file. Rather, the entire file must be read into memory first.

## 8-2 SEQUENTIAL FILES (CONT)

The previous lesson discussed all the basic commands required to create and use a sequential file. This lesson will introduce some additional command words which will allow the user to manipulate those files.

APPEND allows the user to add information to an existing file. The new data is placed immediately after the old. The APPEND command is used in place of the OPEN command. (Since the OPEN command always starts at the beginning of a file, any attempt to WRITE new information using the OPEN command will write over and erase old information.)

```
ie. 10 PRINT CHR$(4); "APPEND STEEL"
    20 PRINT CHR$(4); "WRITE STEEL"
```

There is no way to remove part of a file if it is no longer needed. Instead, the user should transfer the good information to a temporary file, DELETE the old file, then RENAME the temporary file with the old name. The RENAME command can be used directly in immediate mode or with the CTRL-D in program mode. After the RENAME command, type the temporary file name, comma, and the new file name.

```
ie. 10 PRINT CHR$(4); "RENAME TEMP,STEEL"
```

The RENAME command can be used for program names as well as file names.

FOR...NEXT statements can be used to read data from a file when the exact amount of data is known; however, sometimes it is easier to instruct the computer to continue reading data until the end of the file is reached.

```
ie. 10 PRINT CHR$(4); "OPEN CLASS"
    20 PRINT CHR$(4); "READ CLASS"
    30 I=I+1
    40 INPUT C$(I), SC(I)
    50 GOTO 30
```

This program will instruct the computer to read an entire file. However, when the end of the file is reached, an OUT OF DATA error will occur and execution will stop. To avoid this problem, the ONERR GOTO <ON ERROR GOTO> command is used. This statement tells the computer what to do when an error is encountered.

```
ie. 05 ONERR GOTO 60
    <05 ON ERROR GOTO 60>
    10 PRINT CHR$(4); "OPEN CLASS"
    20 PRINT CHR$(4); "READ CLASS"
    30 I=I+1
    40 INPUT C$(I), SC(I)
```

```
50 GOTO 30  
60 PRINT CHR$(4); "CLOSE CLASS"
```

If an error occurs after the ONERR GOTO <ON ERROR GOTO> command, execution branches to the line number specified. It is important, therefore, to put the ONERR <ON ERROR> command early in the program.

<The TRS80 requires the command word RESUME at the end of any error subroutine. This command can be of the following forms:

- 1) RESUME - continue at the line in which the error occurred
- 2) RESUME 0 - continue at the line in which the error occurred
- 3) RESUME 100 - continue at line 100
- 4) RESUME NEXT - continue at the line following the statement in which the error occurred.>

## 9-1 RANDOM ACCESS FILES

A random access file is similar to a filing system composed of a group of drawers. The information in each "drawer" is called a record. One record can hold one or several pieces of data, but all records within a file must be the same length. A random access file is more flexible than a sequential file in that any record can be accessed individually.

The random access file commands are very similar to the sequential file commands with a few exceptions. Random access files must have the record length specified in the OPEN statement. To specify the length, type an L followed by the number of characters in one record.

```
ie. 10 PRINT CHR$(4); "OPEN FILENAME, L50"
```

This statement specifies that each record in the file can hold up to 50 characters.

The WRITE command sends information to a specific record in the file. The file name must be followed by a comma and an R to denote random access.

```
ie. 20 PRINT CHR$(4); "WRITE FILENAME,R"; N
```

In the preceding statement, N stands for the record number. N can be a number or a variable. The following program creates a file called ROSTER and stores three names and social security numbers.

```
ie. 10 PRINT CHR$(4); "OPEN ROSTER, L25"
    20 FOR I=1 TO 3
    30 INPUT "NAME: "; N$(I)
    40 INPUT "SOCIAL SECURITY #"; SS$(I)
    50 NEXT I
    60 FOR I=1 TO 3
    70 PRINT CHR$(4); "WRITE ROSTER,R"; I
    80 PRINT N$(I); SS$(I)
    90 NEXT I
    100 PRINT CHR$(4); "CLOSE ROSTER"
```

This file would be structured as follows:

```
Record 1   SAM SMITH492-60-1629
Record 2   DARLA JONES421-92-8517
Record 3   JERRY ANDERSON467-80-9561
```

Note that record 3 is exactly 25 characters long. If Jerry were replaced by Michael, the last two characters of the social security number would have been cut off. (Blank spaces count as one character.)

Rather than dumping the entire file into memory, the user can access one record at a time.

```
ie. 10 PRINT CHR$(4); "READ FILENAME,R"; N
```

Again, the file name must be followed by a comma and an R. In this example, only record N is read.

## APPENDIX II

CIVIL ENGINEERING APPLE LIBRARY  
PROGRAM DESCRIPTIONS & USER INSTRUCTIONS!1j

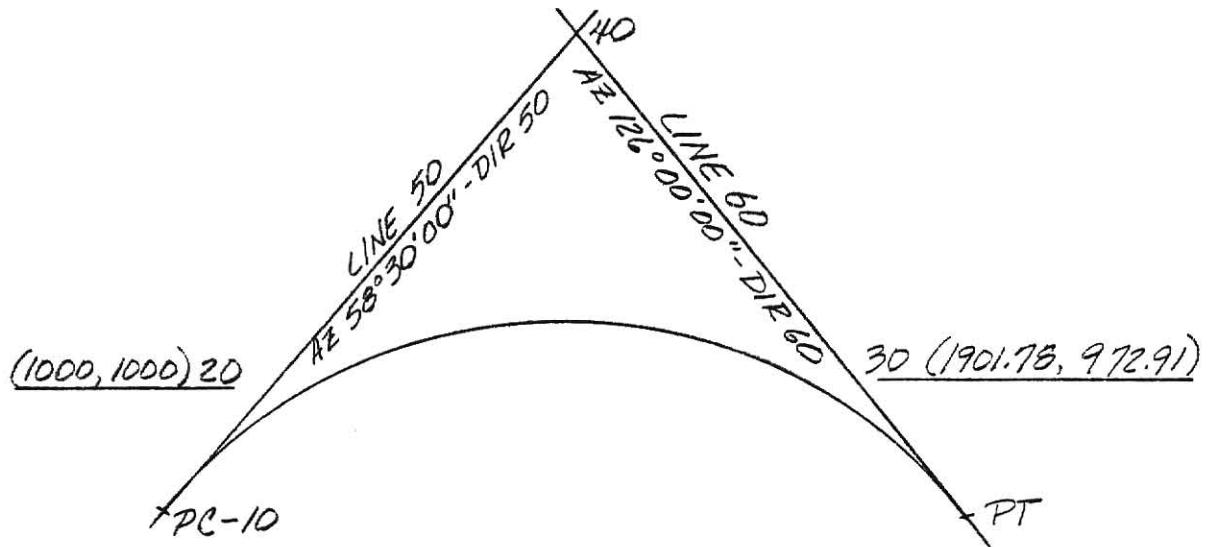
Table of Contents	Page
COGO	71
Concrete Beam	76
Earthwork	78
Gas Flow	80
Oxygen Sag Curve	82
Settlement	84
Simultaneous Equations	86
Spiral	87
Traverse	90
Truss Analysis	93

## COGO

This program will solve surveying problems dealing with points, lines, curves, etc as used in Route Location & Design classes. The COGO diskette must be inserted into Drive 1 and the COGO storage diskette or a blank diskette must be inserted into Drive 2.

## Example:

Find the distance from PC on Curve 70 to Point 40.



## Run:

IS THIS THE FIRST TIME YOU HAVE USED THE DISK IN DRIVE TWO FOR STORAGE? Y OR N. ?Y

THIS DISK WILL BE INITIALIZED TO STORE DATA FOR SEVERAL PROJECTS. AT THE END OF THIS RUN, YOU WILL HAVE THE OPTION OF SAVING YOUR DATA FOR USE. PLEASE WAIT A FEW MINUTES FOR THE INITIALIZATION.

## INFORMATION ON STORAGE

## EACH STORAGE DISK WILL STORE:

1000 POINTS  
600 DISTANCES  
600 ANGLES  
600 DIRECTIONS  
600 LINES  
200 CURVES

THE ELEMENTS FOR EACH PROJECT (POINTS, CURVES, ETC) MUST BE NUMBERED SEQUENTIALLY. EX. - IF PROJECT #1 CONTAINS 60 POINTS, THEY MUST BE NUMBERED 1 TO 60. (PRESS ENTER TO CONTINUE)

THEY MAY BE STORED IN ANY ORDER; IE. PT 1 THEN PT 42.



IF PROJ. #2 HAS 10 POINTS, THEY MUST BE NUMBERED 61 TO 71.  
EACH TIME YOU BEGIN A PROJECT, YOU WILL GET A DIRECTORY OF  
WHICH NUMBERS HAVE BEEN USED ON EACH PROJECT.

### MENU

YOU MAY DO ONE OF THE FOLLOWING:

- (1) USE STORE AND/OR LOCATE COMMANDS
- (2) START A NEW PROJECT
- (3) END A PROJECT

?1

IN THIS PART OF THE PROGRAM, YOU WILL BE ABLE TO USE THE  
FOLLOWING COMMANDS BY TYPING THE ABBREVIATIONS IN PARENTHESIS.

(STO): STORE OBJECT

(LOC): LOCATE POINT

IN ADDITION, YOU CAN DO THE FOLLOWING:

SEE MAIN MENU BY TYPING (MAIN)

SEE THIS MENU BY TYPING (RETURN)

RECALL WHAT YOU HAVE DONE BY TYPING (RECALL)

WHAT WOULD YOU LIKE TO DO NOW?

(STO)RE, (LOC)ATE, (OUT)PUT, (STA)TION,

(MAIN), (RETURN), (RECALL)

?STO

STORE WHAT?: (P)OINT, (D)ISTANCE, (A)NGLE,  
(L)INE, (C)URVE, (DI)RECTION

?P

WHAT IS THE NUMBER OF POINT? ?20

WHAT IS THE X COORDINATE OF POINT 20? ?1000

Y COORDINATE? ?1000

POINT 20 IS NOW STORED.

WHAT WOULD YOU LIKE TO DO NOW? ?STO

STORE WHAT? ?P

WHAT IS THE NUMBER OF POINT? ?30

WHAT IS THE X COORDINATE OF POINT 30? ?1901.78

Y COORDINATE? ?972.91

POINT 30 IS NOW STORED.

WHAT WOULD YOU LIKE TO DO NOW? ?STO

STORE WHAT? ?DI

WHAT IS THE NUMBER OF THIS DIRECTION? ?50

ARE YOU GOING TO ENTER:

THE DIRECTION AS:

(1) A BEARING

(2) AN AZIMUTH

(3) ANOTHER DIRECTION PLUS OR MINUS AN ANGLE

?2

ARE YOU GOING TO USE:

(1) DEGREES, MINUTES, SECONDS

(2) DECIMALS OF A DEGREE

?1

ENTER DIRECTION (D,M,S)  
(DEGREES, MINUTES, SECONDS) ?58,30,00  
DIRECTION 50 IS STORED.

WHAT WOULD YOU LIKE TO DO NOW? ?STO  
STORE WHAT? ?DI  
WHAT IS THE NUMBER OF THIS DIRECTION? ?60  
ARE YOU GOING TO ENTER THE DIRECTION AS: ?2  
ARE YOU GOING TO USE: ?1  
ENTER DIRECTION (D,M,S) ?126,00,00  
DIRECTION 60 STORED.

WHAT WOULD YOU LIKE TO DO NOW? ?STO  
STORE WHAT? ?L  
WHAT IS THE NUMBER OF THIS LINE? ?50  
ARE YOU GOING TO STORE THIS LINE?  
(1) GIVING A POINT AND DIRECTION  
(2) GIVING TWO POINTS  
(3) GIVING PARALLEL LINE AND OFFSET  
?1  
WHAT IS THE NUMBER OF THE POINT? (THIS POINT MUST HAVE  
ALREADY BEEN STORED). ?20  
HAS THE DIRECTION ALREADY BEEN STORED? (Y/N) ?Y  
WHAT IS THE NUMBER OF THE DIRECTION? ?50  
LINE 50 STORED.

WHAT WOULD YOU LIKE TO DO NOW? ?STO  
STORE WHAT? ?L  
WHAT IS THE NUMBER OF THIS LINE? ?60  
ARE YOU GOING TO STORE THIS LINE: ?1  
WHAT IS THE NUMBER OF THE POINT? ?30  
HAS THE DIRECTION ALREADY BEEN STORED? (Y/N) ?Y  
WHAT IS THE NUMBER OF THE DIRECTION? ?60  
LINE 60 STORED.

WHAT WOULD YOU LIKE TO DO NOW? ?LOC  
WITH THIS COMMAND YOU MAY DO THE FOLLOWING:  
(1) LOCATE FROM POINT  
(2) LOCATE ON OBJECT  
(3) LOCATE INTERSECT  
WHAT IS THE NUMBER OF THE POINT YOU WISH TO LOCATE? ?40  
WHICH OF THE ABOVE DO YOU WISH TO DO? (1,2,OR 3) ?3  
WHAT IS THE NUMBER OF THE LINE ONE YOU ARE LOCATING ON? ?50  
WHAT IS THE NUMBER OF THE LINE TWO YOU ARE LOCATING ON? ?60  
POINT 40 NOW STORED.

WHAT WOULD YOU LIKE TO DO NOW? ?STO  
STORE WHAT? ?C  
WHAT IS THE NUMBER OF THE CURVE? ?70  
TO STORE A CURVE, YOU MUST GIVE ONE DIMENSION OUT OF EACH OF  
THE FOLLOWING THREE SECTIONS (A,B,& C). PRESS RETURN TO SEE  
THE SECTIONS.

#### SECTION A

(1) PC POINT NUMBER AND DIRECTION BACK  
 (2) PI POINT NUMBER AND DIRECTION BACK  
 (3) POINT BACK AND PI POINT NUMBER  
 WHICH DIMENSION OF SECTION A? ?2

SECTION B

(1) RADIUS LENGTH  
 (2) DEGREE OF CURVE  
 (3) LENGTH OF CURVE  
 (4) TANGENT LENGTH  
 WHICH DIMENSION OF SECTION B? ?2

SECTION C

(1) DIRECTION AHEAD  
 WHICH DIMENSION OF SECTION C? ?1

WHAT IS THE NUMBER OF THE PI? ?40  
 WHAT IS THE NUMBER OF THE DIRECTION BACK? ?50  
 IS THE DIRECTION BACK TOWARDS THE PI? ?Y  
 IS THE DEGREE OF CURVE IN (1) DEGREES, MINUTES, SECONDS OR (2)  
 DECIMALS OF A DEGREE? ?1  
 ENTER DEGREE IN D,M,S. ?4,30,00  
 WHAT IS THE NUMBER OF THE DIRECTION AHEAD? ?60  
 IS THE DIRECTION AHEAD AWAY FROM THE PI? ?Y  
 WHAT NUMBER DO YOU WISH TO ASSIGN TO THE PC? ?10  
 CURVE 72 NOW STORED.

WHAT WOULD YOU LIKE TO DO NOW? ?STO  
 STORE WHAT? ?D  
 WHAT IS THE NUMBER OF THIS DISTANCE? ?10  
 DO YOU WANT TO GIVE:  
 (1) THE MAGNITUDE OF THE DISTANCE  
 (2) TWO POINTS (FIND THE DISTANCE)  
 ?2  
 WHAT ARE THE TWO POINTS? (P1,P2). ?10,20  
 DISTANCE 10 IS STORED.

WHAT WOULD YOU LIKE TO DO NOW? ?OUT  
 WHICH OF THE FOLLOWING DO YOU WANT PRINTED?  
 (1) POINT INFORMATION  
 (2) DISTANCE  
 (3) ANGLE  
 (4) DIRECTION  
 (5) LINE  
 (6) CURVE  
 ?6

WHAT IS THE NUMBER OF THE CURVE? ?70  
 DO YOU WANT A PRINTED OUTPUT? ?Y

INFORMATION ON CURVE 70:

COORDINATES - PC: X=743.569958 Y=842.859466  
 COORDINATES - PT: X=2157.22722 Y=787.31672  
 COORDINATES - PI: X=1468.95483 Y=1287.37589  
 RADIUS = 1273.23954 FEET  
 DELTA ANGLE = 67 DEGREES, 30 MINUTES, 1.07288361 E-03 SECONDS

DEGREE OF CURVE = 4 DEGREES, 30 MINUTES, 6.03497028 E-05  
SECONDS  
TANGENT DISTANCE = 850.751466 FEET  
LENGTH OF CURVE = 1500 FEET  
EXTERNAL DISTANCE = 258.072637 FEET

DO YOU WANT MORE OUTPUT? ?Y  
WHICH OF THE FOLLOWING DO YOU WANT PRINTED? ?2  
WHAT IS THE NUMBER OF THE DISTANCE? ?10  
DO YOU WANT A PRINTED OUTPUT? ?Y

THE MAGNITUDE OF DISTANCE 10 IS: 300.748257 FEET

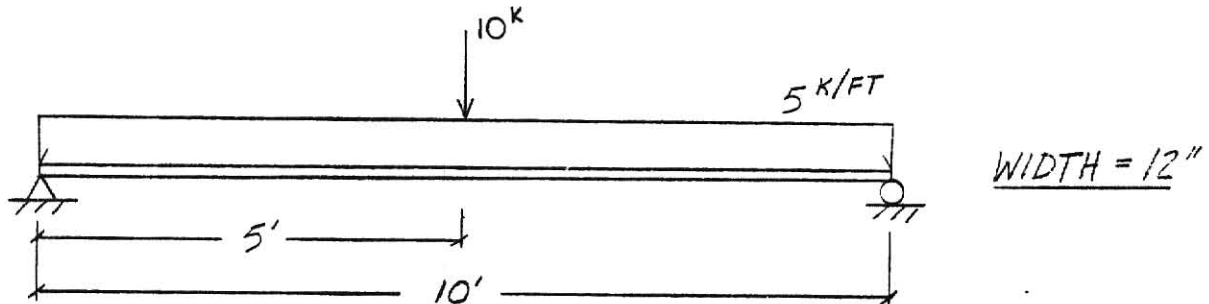
DO YOU WANT MORE OUTPUT? ?N

## CONCRETE BEAM

This program will design a simply supported concrete beam for maximum positive and negative steel using Ultimate Strength Design as used in Concrete Design classes. The theory is based on a balanced force system between the concrete in compression and the steel in tension. All cases can be solved except for a cantilever support.

## Example:

Design the following concrete beam using a concrete strength of 4000 psi and a steel strength of 60,000 psi. The beam dead weight is 0.5 k/ft. Use #7 reinforcing bar.



## Run:

THIS PROGRAM DESIGNS CONCRETE BEAMS FOR CROSS SECTIONAL DIMENSIONS AND REINFORCEMENT STEEL FOR THE FOLLOWING SIMPLY SUPPORTED LOADED CONDITIONS:

1. SUPPORTS AT ENDS
2. OVERHANGING END(S)
3. LIVE POINT AND/OR DISTRIBUTED LOADS

PLEASE FOLLOW THE INSTRUCTIONS IN THE USER'S GUIDE TO RUN. THIS IS TO BE USED ONLY FOR TUTORIAL PURPOSES. ITS USE IN PROFESSIONAL PRACTICE IS PROHIBITED.

ENTER SPAN LENGTH: ?10

ENTER POSITION OF SUPPORTS FROM LEFT: ?0,10

ENTER VALUE OF DEAD LOAD OF BEAM (K/FT): ?0.5

HOW MANY LIVE LOAD DISTRIBUTED SEGMENTS?: ?1

ENTER END POINTS FROM LEFT, AND VALUE, (K/FT) FOR MAX POS MOMENT: #,#,#: ?0,10,5

HOW MANY LIVE POINT LOADS: ?1

ENTER POSITION FROM LEFT, AND VALUE, (K), FOR MAX POS MOMENT:  
#,: 75,10

PLEASE WAIT AS V & M VALUES ARE COMPUTED.

VL = 86 K                      VR = 86 K  
MX M = 235.9 K-FT

ENTER CONCRETE STRENGTH (PSI): 74000  
ENTER STEEL STRENGTH (PSI): 760000

ARE CROSS-SECTIONAL DIMENSIONS FINALIZED (Y/N): ?N

WIDTH X (EFF DEPTH)SQRD = 2019.51917 IN CBD  
SELECT A WIDTH, B: ?12  
EFFECTIVE DEPTH = 12.5 IN  
TOTAL DEPTH = 15.5 IN  
VALUE OF REVISED DEAD LOAD IS: .19 K/FT  
MX M = 152.2 K-FT

ARE CROSS-SECTIONAL DIMENSIONS FINALIZED? (Y/N) ?Y

AREA OF STEEL REQUIRED = 3.2 SQ IN  
SELECT BARS FROM THE FOLLOWING TABLE:

ENTER BAR NUMBER YOU WISH: ?7  
CHOICES: #4-#11, #14, #18

TYPE - #7 BAR  
AREA OF STEEL REQUIRED = 3.2 SQ IN

NUMBER OF BARS	CROSS-SECT AREA (SQ IN)
2	1.2
3	1.8
4	2.41
5	3.01
6	3.61
7	4.21
8	4.81
9	5.41
10	6.01
11	6.61
12	7.22
13	7.82
14	8.42

HOW MANY BARS DO YOU WISH TO FIT? (ENTER '1' IF ANOTHER BAR IS DESIRED) ?6

STEEL CAN BE PLACED IN 2 ROWS OF 3 EACH. TOTAL AREA OF STEEL = 3.61 SQ IN.

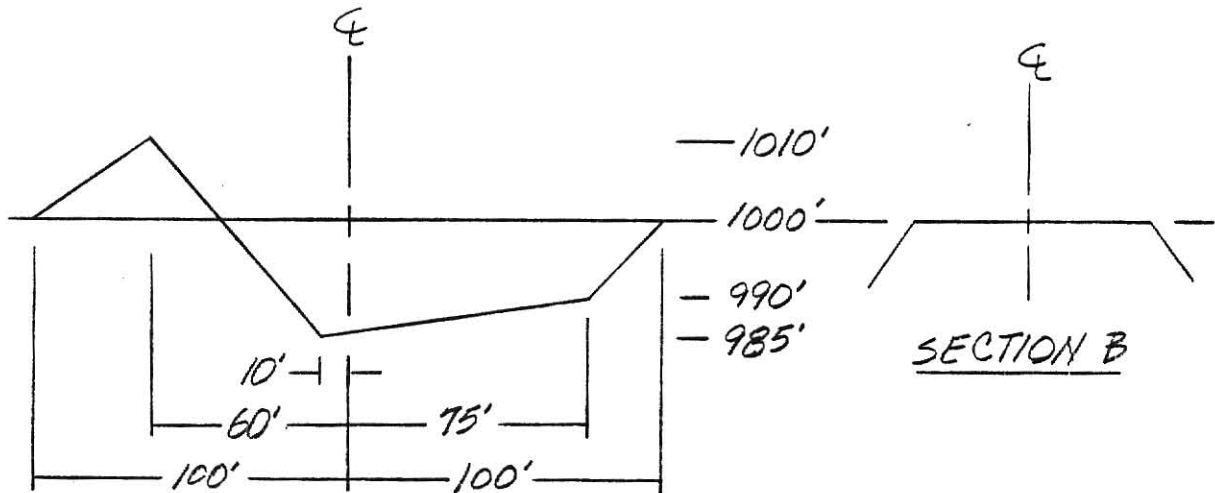
NOW THAT STEEL REQUIREMENTS ARE COMPUTED, THE FOLLOWING GRAPHIC SHOWS SOME BASIC ACI CROSS-SECTIONAL REQUIREMENTS FOR PLACEMENT OF THE STEEL IN THE BEAM.

## EARTHWORK

This program will calculate the cut and fill end areas for a roadway cross section as used in Route Location and Design classes. If the slope of the land is greater than 1:2, the program will inform the user that the terrain is unfit for a roadway.

## Example:

For the following topographical cross section, superimpose the road section B and calculate the amount of cut and fill.



Run:

ENTER THE NUMBER OF SURFACE ELEVATION POINTS. ?5

THE POINTS SHOULD BE ENTERED FROM LEFT TO RIGHT. I WILL ASK THE QUESTIONS. ALL YOU HAVE TO DO IS ANSWER THEM.-

DUE TO RESTRICTIONS IN THE PROGRAM, YOU MUST INPUT TERRAIN PROFILE 100 FEET IN BOTH DIRECTIONS FROM THE CENTERLINE OF THE PROPOSED ROADWAY.

IS THE POINT LEFT OF CENTERLINE? ?Y

HOW FAR TO THE LEFT OF CENTERLINE IS THE POINT? ?100

WHAT IS THE ELEVATION AT THIS DISTANCE FROM CENTERLINE? ?1000

IS THE POINT LEFT OF CENTERLINE? ?Y

HOW FAR TO THE LEFT OF CENTERLINE IS THE POINT? ?60

WHAT IS THE ELEVATION AT THIS DISTANCE FROM CENTERLINE? ?1010

IS THE POINT LEFT OF CENTERLINE? ?Y

HOW FAR TO THE LEFT OF CENTERLINE IS THE POINT? ?10

WHAT IS THE ELEVATION AT THIS DISTANCE FROM CENTERLINE? ?985

IS THE POINT LEFT OF CENTERLINE? ?N

HOW FAR TO THE RIGHT OF CENTERLINE IS THE POINT? ?75

WHAT IS THE ELEVATION AT THIS DISTANCE FROM CENTERLINE? ?990

IS THE POINT LEFT OF CENTERLINE? ?N

HOW FAR TO THE RIGHT OF CENTERLINE IS THE POINT? ?100

WHAT IS THE ELEVATION AT THIS DISTANCE FROM CENTERLINE? ?1000

BIGY = 1000 AT XBIG = 200

LOWY = 990 AT XLOW = 175

WHICH CROSS SECTION WOULD YOU LIKE TO SUPERIMPOSE ON THE  
GROUND SURFACE? (A,B,C,D) ?B

AT WHAT ELEVATION WOULD YOU LIKE THE CENTERLINE OF THE  
ROADWAY? ?1000

DO YOU WANT TO CHANGE THE ELEVATION OF THE CENTERLINE? ?N

DO YOU WANT TO CHANGE CROSS SECTIONS? ?N

PLEASE WAIT WHILE I APPROXIMATE THE END AREAS.

FILL AREA = 1615

CUT AREA = 2153



## GAS FLOW

This program solves for the properties of a certain gas flowing through a pipeline as used in Hydraulics or Sanitary Engineering classes. The properties include pressure, temperature, density, and velocity and are calculated at one mile intervals for a straight pipe of uniform diameter with constant flow.

### Example:

A steel pipeline is to carry 80,000 cu ft/day of methane gas. The pipeline is 6" in diameter and 6 miles long. The initial gas temperature is 70°F and the initial gas pressure is 50 psia. Find the pressure of the gas at one mile intervals along the pipeline.

### Run:

THIS PROGRAM IS CONCERNED WITH THE TRANSPORTATION OF SEVERAL GASES THROUGH PIPE LINES MADE OF DIFFERENT MATERIALS.

THE INPUTS CONSIST OF INITIAL GAS AND PIPE LINE PROPERTIES. THE OUTPUTS CONSIST OF GAS PROPERTIES AT ONE MILE INTERVALS ALONG THE PIPE LINE.

THERE ARE THREE TYPES OF PIPES WHICH ARE COMMONLY USED TO TRANSPORT GASES:

- 1) CAST IRON
- 2) WROUGHT IRON
- 3) STEEL

ENTER THE NUMBER FOR THE TYPE OF PIPE TO BE USED (1-3). ?3

WHICH ONE OF THE FOLLOWING GASES IS TO BE TRANSPORTED THROUGH THIS PIPE LINE:

- 1) WATER STEAM
- 2) CARBON DIOXIDE
- 3) METHANE
- 4) DRY AIR
- 5) CHLORINE
- 6) OXYGEN
- 7) HYDROGEN

ENTER THE NUMBER FOR THE GAS TO BE TRANSPORTED (1-7). ?3

THE INPUTS FOR THIS PROGRAM INCLUDE PIPELINE INFORMATION AS WELL AS THE INITIAL GAS PROPERTIES.

HOW MUCH GAS IS GOING TO BE TRANSPORTED? (CU FT/DAY) ?80000

HOW LONG IS THE PIPELINE? (MILES) (1-100) ?6

WHAT IS THE PIPE NOMINAL DIAMETER? (INCHES) ?6

WHAT IS THE INITIAL TEMPERATURE OF THE GAS? (FAHRENHEIT) ?70

WHAT IS THE INITIAL PRESSURE? (PSIA) ?50

### MAIN MENU

- 1) COMPUTE FLUID PRESSURES AT ONE MILE INTERVALS

- 2) COMPUTE FLUID TEMPERATURES AT ONE MILE INTERVALS
- 3) COMPUTE FLUID DENSITIES AT ONE MILE INTERVALS
- 4) COMPUTE FLUID VELOCITIES AT ONE MILE INTERVALS
- 5) EXIT THE PROGRAM

ENTER THE NUMBER OF THE ROUTINE TO BE PERFORMED (1-5). ?1

THIS ROUTINE COMPUTES THE PRESSURE DROP AT ONE MILE INTERVALS  
ALONG THE PIPE LINE.

THE LENGTH OF THE PIPE LINE IS 6 MILES.

THE DIAMETER OF THE PIPE IS 6 INCHES.

THE GAS FLOW THROUGH THE PIPE IS 80000 CUBIC FT PER DAY.

DISTANCE (MILES)	PRESSURE (PSIA)
0	50
1	49.9941308
2	49.988234
3	49.9823088
4	49.9763554
5	49.9703737
6	49.964358

WHAT WOULD YOU LIKE TO DO NEXT?

- 1) GO BACK TO THE MAIN MENU
- 2) EXIT THE PROGRAM

ENTER THE NUMBER OF YOUR CHOICE (1-2). ?2

## OXYGEN SAG CURVE

This program calculates and draws an oxygen sag curve as used in Sanitary Fundamentals classes. The curve shows the lowest dissolved oxygen level. Also calculated are the time and distance downstream at which the critical dissolved oxygen level occurs.

## Example:

For a population of 200,000 and given the following data, determine the time and distance downstream at which the critical dissolved oxygen occurs.

River Temperature = 24°C	Sewage Temperature = 25.5°C
River BOD = 3.6 mg/l	Sewage BOD = 65 mg/l
River Velocity = 1.2 ft/s	Sewage Production = 120 gpcd
River Flow = 250 cfs	Sewage DO = 1.8 mg/l
River Depth = 8 ft	
Percent Saturation = 90	
Deoxygenation Coefficient = 0.5	

## Run:

WOULD YOU LIKE TO KNOW MORE ABOUT BOD AND THE OXYGEN SAG CURVE? (Y OR N) ?N

DO YOU WANT INSTRUCTIONS ON HOW TO RUN THE PROGRAM? (Y OR N) ?N

THIS PROGRAM FIGURES ALL OF THE CRITICAL VALUES YOU WILL NEED WHEN PLOTTING AN OXYGEN SAG CURVE, THEN IT WILL PLOT THE CURVE. PRETTY NIFTY, HUH?

ARE YOU READY TO BEGIN? ?Y

WHAT IS THE TEMPERATURE OF THE RIVER? (IN DEGREES CELSIUS) TYPICAL VALUES RANGE FROM 0 TO 35 DEGREES. ?24

WHAT IS THE TEMPERATURE OF THE SEWAGE? (IN DEGREES CELSIUS) TYPICAL VALUES RANGE FROM 10 TO 35 DEGREES. ?25.5

WHAT IS THE COEFFICIENT OF DEOXYGENATION? TYPICAL VALUES RANGE FROM .25 TO 1. ?0.5

WHAT IS THE VELOCITY OF THE RIVER FLOW? (IN FT/S) TYPICAL VALUES RANGE FROM .1 TO 10. ?1.2

WOULD YOU LIKE TO HAVE THE SEWAGE FLOW (C)ALCULATED OR WOULD YOU LIKE TO (P)ROVIDE IT? ?C

WHAT IS THE POPULATION OF THE CITY? (IN 1000'S) ?200

WHAT IS THE SEWAGE PRODUCTION IN GALLONS PER CAPITA PER DAY? (THE AVERAGE RATE IS 100 GPCD). ?120

THE SEWAGE FLOW IS 37.128 CFB.

WHAT IS THE FLOW OF THE RIVER? (IN CFS) TYPICAL VALUES ARE GREATER THAN 1. ?250

WHAT IS THE PERCENT SATURATION OF THE RIVER WITH OXYGEN? VALUES USUALLY RANGE FROM 50 TO 100. ?90

WHAT IS THE DISSOLVED OXYGEN OF THE SEWAGE? (IN MILLIGRAMS/LITER) VALUES USUALLY RANGE FROM 0 TO 5. ?1.8

THE DISSOLVED OXYGEN OF THE SEWAGE-RIVER MIXTURE IS 6.91705581 MILLIGRAMS PER LITER.

THE TEMPERATURE OF THE SEWAGE-RIVER MIXTURE IS 24.1939623 DEGREES CELSIUS.

THE INITIAL OXYGEN DEFICIT IS 1.58384985 MILLIGRAMS PER LITER.  
WHAT IS THE BOD OF THE RIVER? (IN MILLIGRAMS/LITER) TYPICAL VALUES RANGE FROM 0 TO 10. ?3.6

WHAT IS THE DEPTH OF THE RIVER? (IN FT) THE DEPTH USUALLY RANGES FROM 1 TO 20 FEET. ?8

WHAT IS THE BOD OF THE SEWAGE? (IN MILLIGRAMS/LITER) SEWAGE BOD'S RANGE FROM ABOUT 5 TO 200. ?65

THE INITIAL BOD OF THE SEWAGE-RIVER MIXTURE IS 11.5395224 MILLIGRAMS PER LITER.

THE CRITICAL TIME IS 1.23895596 DAYS.

THE MINIMUM DISSOLVED OXYGEN 4.16169412 MILLIGRAMS PER LITER.

THE CRITICAL DEFICIT IS 4.33921154 MILLIGRAMS PER LITER.

THE DISTANCE DOWNSTREAM AT WHICH THE CRITICAL DISSOLVED OXYGEN OCCURS IS 128454.954 FEET.

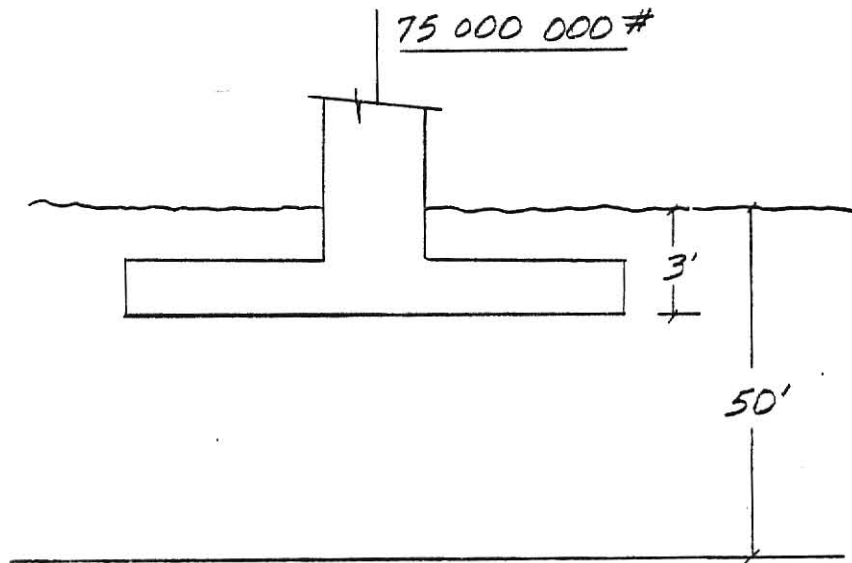
WOULD YOU LIKE A PRINTOUT OF THE CALCULATED VALUES? ?N

## SETTLEMENT

This program will compute settlement as described in beginning Soil Mechanics classes. Any number of soil layers and soil types may be input. The water table may be within a soil layer.

## Example:

Calculate the settlement for one layer of silty clay with specific gravity 2.65. The compression coefficient is 0.125 and gamma dry is 90. The water table is below the layer.



## Run:

THIS PROGRAM IS DESIGNED TO COMPUTE SETTLEMENT.  
THROUGHOUT THIS PROGRAM YOU WILL BE ASKED CERTAIN QUESTIONS.  
THESE QUESTIONS WILL BE REQUESTS FOR SUCH DATA AS:

1. DEPTH OF SOIL LAYER.
2. DEPTH TO WATER TABLE.
3. VOID RATIO.
4. AREA OF FOOTING.
5. IS THE LAYER SAND?

THESE ARE BUT A FEW PIECES OF DATA THAT WILL BE NEEDED TO  
SUCCESSFULLY COMPUTE SETTLEMENT.

JUST TO REFRESH YOUR MEMORY OF WHAT WE ARE TALKING ABOUT...

PRESS SPACE BAR TO CONTINUE

YOU MAY FEEL THAT YOU MIGHT WANT SOME HELP WITH OPERATING THE  
APPLE.

IF YOU FEEL LOST PUSH THE KEY LABELED 'Y' TO CONTINUE.  
OTHERWISE HIT THE 'RETURN KEY.'

THE FOOTING IS ASSUMED TO BE A SQUARE FOOTING WITH LENGTH  
EQUAL TO WIDTH.

WHAT IS THE LENGTH OR WIDTH OF THE FOOTING? ?135

WHAT IS THE LOADING ON THE FOOTING EXPRESSED IN POUNDS?

?75000000

WHAT IS THE DEPTH OF THE FOOTING? ?3

WE WILL NOW BEGIN THE PROCESS OF COMPUTING SETTLEMENT. HOW  
MANY DISTINCT SOIL LAYERS ARE THERE? ?1

NOTE: WHEN ASKED ABOUT THE DEPTH OF THE FIRST LAYER GIVE ITS  
DEPTH AS THE TOTAL DEPTH OF THE LAYER MINUS THE DEPTH OF THE  
FOOTING.

THIS DATA CONCERNS SOIL LAYER #1.

WHAT IS THE DEPTH OF THIS LAYER? ?47

WHAT IS THE COMPRESSION COEFFICIENT (C SUB C) FOR THIS LAYER?  
(LIMITS ARE ZERO TO ONE)? ?125

WHAT IS GAMMA DRY FOR THIS LAYER (LIMITS ARE 70 TO 140)? ?90

WHAT IS THE SPECIFIC GRAVITY FOR THIS LAYER (LIMITS ARE 2.5 TO  
2.8)? ?2.65

THE CALCULATED VOID RATIO IS .837333333.

THIS COMPLETES THE DATA PERTINENT TO THE INDIVIDUAL SOIL  
LAYERS.

THESE QUESTIONS REFER TO LAYER #1.

IS THIS LAYER SAND? (Y/N) ?N

IS THE WATER TABLE LOCATED IN THIS LAYER? (Y/N) ?N

THE LAYER IN QUESTION IS NOT SAND AND THE WATER TABLE IS NOT  
LOCATED IN IT.

THE NUMBER OF 2 FOOT INCREMENTS IS 23.

THERE IS 1 INCREMENT OF 1 FOOT.

NOW THAT WE KNOW THE NUMBER OF 2 AND 1 FOOT INCREMENTS WE CAN  
CONTINUE.

PAUSE HERE TO GIVE THE COMPUTER TIME TO CALCULATE SETTLEMENT.  
HAVE PATIENCE!

THE TOTAL SETTLEMENT THRU THIS LAYER IS 12.5036139 INCHES.

## SIMULTANEOUS EQUATIONS

This program solves a system of linear equations as used in Mathematics and Design classes. The number of unknown coefficients in each equation must equal the number of equations being solved.

Example:

Solve the following system of equations:

$$\begin{aligned} A + 2B + 3C &= 4 \\ 3A + 6B &= 1 \\ -3A + 4B - 2C &= 0 \end{aligned}$$

Run:

DO YOU WISH A COPY OF THE OUTPUT FROM THE PRINTER? (Y/N) ?N

NUMBER OF EQUATIONS? ?3

COEFFICIENT MATRIX:

EQUATION #1

COEFFICIENT 1 = 1  
COEFFICIENT 2 = 2  
COEFFICIENT 3 = 3  
CONSTANT = 4

EQUATION #2

COEFFICIENT 1 = 3  
COEFFICIENT 2 = 6  
COEFFICIENT 3 = 0  
CONSTANT = 1

EQUATION #3

COEFFICIENT 1 = -3  
COEFFICIENT 2 = 4  
COEFFICIENT 3 = -2  
CONSTANT = 0

VALUE OF DETERMINANT = 90

X(1) = -.356

X(2) = .344

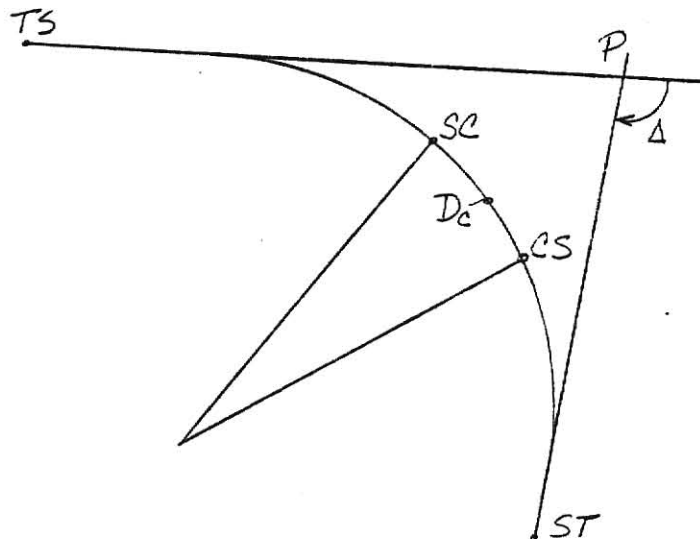
X(3) = 1.222

## SPIRAL

This program will design a transition spiral curve connecting two perpendicular roadways as used in Route Location and Design classes. The output is in the form of a set of field notes that can be used when laying out the curve. The output data can also be used in conjunction with the COGO program. The field notes are calculated using either station intervals or N-chords.

## Example:

Calculate a set of field notes to lay out a 500' long spiral curve using the Chord Method with 11 chords. The TS is at Station 3221, the delta angle is  $48^{\circ}48'59''$ , the degree of the simple curve is  $6^{\circ}00'00''$ , and station intervals are 50'.



## Run:

THIS PROGRAM CALCULATES FIELD NOTES FOR SET-UP AT THE TS, SC, AND CS; AND X,Y COORDINATES FOR A SPIRALED CURVE. ALSO IT WILL FIND THE POINT OF INTERSECTION BETWEEN THE SPIRAL AND A GIVEN LINE.

## INPUTS REQUIRE:

LENGTHS IN FEET

ANGLES IN DEGREES, MINUTES, SECONDS

DIRECTIONS AS NORTH AZIMUTHS

WHAT IS THE LENGTH OF SPIRAL IN FEET? 7500

WHAT IS THE STATION OF THE TS? 73221

WHAT IS THE DELTA ANGLE? DEGREES, MINUTES, SECONDS 748, 48, 59

WHAT IS THE DEGREE OF CURVE FOR THE SIMPLE CURVE?

DEGREES, MINUTES, SECONDS 76, 0, 0



## MENU

(1) X,Y COORDINATES  
 (2) FIELD NOTES  
 (3) INTERSECTION OF LINE WITH SPIRAL  
 (4) END PROGRAM  
 ENTER NUMBER: ?2

WOULD YOU LIKE THE SPIRAL CALCULATED USING STATION (S) OR  
 CHORD (C) METHOD? ?C  
 HOW MANY CHORDS? ?11

WHAT STATION INTERVALS ON THE CURVE? (IN FEET) ?50

SEE PRINTER FOR FIELD NOTES.

## SPIRAL DATA:

STATION TS = 3221  
 STATION SC = 3721  
 STATION CS = 4034.60648  
 STATION ST = 4535.60648

## ANGLES IN DEGREES, MINUTES, SECONDS

DELTA ANGLE = 48 48 59  
 SPIRAL ANGLE = 15 0 0  
 LENGTH OF SPIRAL = 500 FEET  
 TANGENT = 687.707948 FEET  
 EXTERNAL = 105.673509 FEET  
 LONG TANGENT = 334.537966 FEET  
 SHORT TANGENT = 167.762325 FEET  
 P = 10.8816453 FEET  
 K = 249.429946 FEET

## CIRCULAR CURVE DATA:

## ANGLES IN DEGREES, MINUTES, SECONDS

DEGREE OF CURVE = 6 0 0  
 LENGTH OF CURVE = 313.606482 FEET  
 RADIUS = 954.929667 FEET

STATION	POINT	ARC	CHORD	DEFL	ANGLE
3221	TS				
45.4545455	45.4545365				
3266.45455	1			00	02 28
45.4545455	45.4544656				
3311.90909	2			00	09 55
45.4545455	45.4543238				
3357.36364	3			00	22 18
45.4545455	45.454111				
3402.81818	4			00	39 40
45.4545455	45.4538273				
3448.27273	5			01	01 58
45.4545455	45.4534727				
3493.72727	6			01	29 15

45.4545455	45.4530471			
3539.18182	7	02	01	28
45.4545455	45.4525506			
3584.63636	8	02	38	39
45.4545455	45.4519832			
3630.09091	9	03	20	46
45.4545455	45.4513448			
3675.54545	10	04	07	50
45.4545455	45.4506356			
3721	11			

STATION	POINT	ARC	CHORD	DEFL	ANGLE
3721	SC				
29.0000007	28.9988859				
3750	1			00	52 12
50	49.9942886				
3800	2			02	22 12
50	49.9942886				
3850	3			03	52 12
50	49.9942886				
3900	4			05	22 12
50	49.9942886				
3950	5			06	52 12
50	49.9942886				
4000	6			08	22 12
34.606481	34.6045873				
4034.60648	7			09	24 29

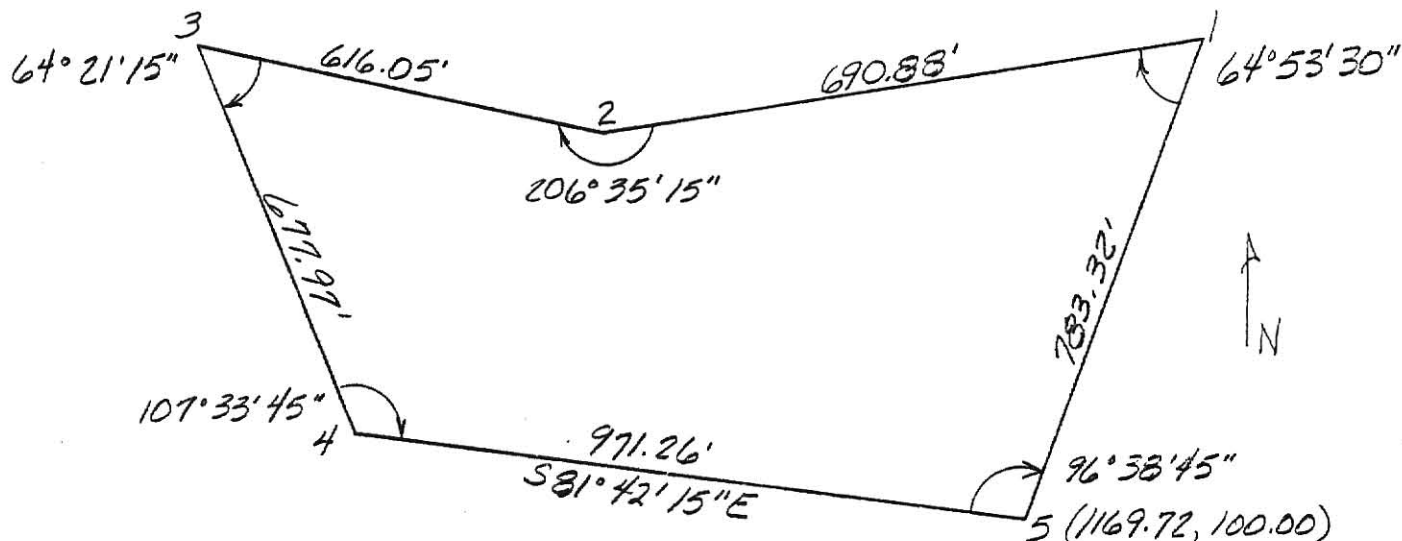
STATION	POINT	ARC	CHORD	DEFL	ANGLE
4034.60648	CS				
45.4545455	45.4506356				
4080.06102	1			01	19 20
45.4545455	45.4513448				
4125.51557	2			02	33 43
45.4545455	45.4519832				
4170.97012	3			03	43 08
45.4545455	45.4524406				
4216.42466	4			04	47 36
45.4545455	45.4530471				
4261.8792	5			05	47 06
45.4545455	45.4534727				
4307.33375	6			06	41 39
45.4545455	45.4538273				
4352.7883	7			07	31 15
45.4545455	45.454111				
4398.24284	8			08	15 53
45.4545455	45.4543238				
4443.69739	9			08	55 35
45.4545455	45.4544656				
4489.15194	10			09	30 20
45.4545455	45.4545365				
4534.60648	11			10	00 10

## TRAVERSE

This program computes a traverse using either the compass or the transit rule. Deflection or clockwise angles and side lengths are the input data. If a direction or coordinates are not specified, course 1-2 will be taken as north and the coordinates of Station 1 will be taken as 10000,10000.

Example:

Compute the following traverse using the compass rule.



Run:

THIS PROGRAM IS FOR SURVEY TRAVERSE SOLUTION. THE PROGRAM WILL ADJUST THE TRAVERSE EITHER BY THE COMPASS RULE OR THE TRANSIT RULE. THE OUTPUT WILL GIVE STATION COORDINATES; COURSE LENGTHS, AZIMUTHS, AND BEARINGS; ALONG WITH THE TRAVERSE AREA; AS DESIRED. SEE PROGRAM INSTRUCTIONS FOR FURTHER DESCRIPTIONS AND USER INFORMATION.

HOW MANY STATIONS DOES YOUR TRAVERSE HAVE? 5  
DO YOU WISH TO ENTER ANGLES AS (D)EFLECTION ANGLES OR  
(C)LOCKWISE MEASURED ANGLES? C

ENTER CLOCKWISE ANGLE AT STATION 1. FIRST ENTER DEGREES, THEN  
MINUTES, THEN SECONDS. 64,53,30  
IS THIS AN (I)NTERIOR OR (E)XTERIOR ANGLE? I

ENTER CLOCKWISE ANGLE AT STATION 2. FIRST ENTER DEGREES, THEN  
MINUTES, THEN SECONDS. 206,35,15  
IS THIS AN (I)NTERIOR OR (E)XTERIOR ANGLE? I

ENTER CLOCKWISE ANGLE AT STATION 3. FIRST ENTER DEGREES, THEN  
MINUTES, THEN SECONDS. 64,21,15  
IS THIS AN (I)NTERIOR OR (E)XTERIOR ANGLE? I

ENTER CLOCKWISE ANGLE AT STATION 4. FIRST ENTER DEGREES, THEN

MINUTES, THEN SECONDS. ?107,33,45

IS THIS AN (I)NTERIOR OR (E)XTERIOR ANGLE? ?I

ENTER CLOCKWISE ANGLE AT STATION 5. FIRST ENTER DEGREES, THEN MINUTES, THEN SECONDS. ?96,38,45

IS THIS AN (I)NTERIOR OR (E)XTERIOR ANGLE? ?I

ENTER DISTANCE FROM STATION 1 TO STATION 2. ?690.88

ENTER DISTANCE FROM STATION 2 TO STATION 3. ?616.05

ENTER DISTANCE FROM STATION 3 TO STATION 4. ?677.97

ENTER DISTANCE FROM STATION 4 TO STATION 5. ?971.26

ENTER DISTANCE FROM STATION 5 TO STATION 1. ?783.32

DO YOU WISH TO ENTER AN ASSUMED DIRECTION FOR 1 COURSE? (Y/N) ?Y

DO YOU WISH TO ENTER ASSUMED DIRECTION AS A (B)EARING OR AN (A)ZIMUTH? ?B

THE ASSUMED COURSE RUNS FROM WHAT STATION TO WHAT STATION? ?4,5

ENTER THE ASSUMED BEARING DIRECTION (N,S,E,W), DEGREES, MINUTES, SECONDS, DIRECTION. ?S,81,42,15,E

DO YOUR TRAVERSE STATIONS INCREASE IN NUMBER IN A COUNTERCLOCKWISE (CC) OR CLOCKWISE (CW) DIRECTION? ?CC

DO YOU WISH TO ADJUST THE TRAVERSE BY THE (C)OMPASS RULE OR THE (T)RANSIT RULE? ?C

DO YOU WISH TO ASSIGN COORDINATES TO ONE POINT? (Y/N) ?Y

ENTER STATION NUMBER YOU ARE ASSIGNING COORDINATES TO. ?5

ENTER X COORDINATE, THEN Y COORDINATE OF STATION NO. 5. ?1169.72,100.00

DO YOU WISH COURSE OUTPUTS AS (A)ZIMUTHS, (B)EARINGS, OR (C)AZIMUTHS AND BEARINGS? ?C

THE MEASURE OF ACCURACY OF THIS TRAVERSE IS 1.3038.

DO YOU WISH A LISTING OF STATION COORDINATES ON THE (S)CREEN, THE (P)RINTER, OR (B)OTH? ?S

STATION NO	X COORDINATE	Y COORDINATE
1	1371.36	857.003
2	691.172	734.979
3	100.003	908.982
4	208.892	239.964
5	1169.72	99.999

DO YOU WISH A LISTING OF COURSE LENGTHS AND ORIENTATIONS ON THE (S)CREEN, THE (P)RINTER, OR (B)OTH? ?S

COURSE	LENGTH	AZIMUTH	BEARING
1-2	691.046	259°49'46"	S79°49'46"W
2-3	616.244	286°24'04"	N73°35'55"W
3-4	677.820	170°45'20"	S 9°14'39"E
4-5	970.968	98°17'16"	S81°42'43"E
5-1	783.398	14°54'55"	N14°54'55"E

DO YOU WISH TO KNOW THE AREA OF THIS TRAVERSE? (Y/N) ?Y  
DO YOU WISH THE AREA LISTED ON THE (S)CREEN, THE (P)RINTER, OR  
(B)OTH? ?S

AREA OF TRAVERSE = 704990.804 SQ.FT.  
AREA OF TRAVERSE = 16.1843619 ACRES

DO YOU WISH A LISTING OF YOUR INPUT DATA? (Y/N) ?N  
WOULD YOU LIKE TO SEE A SKETCH OF THE TRAVERSE? (Y/N) ?Y

WHEN YOU ARE READY TO CONTINUE THE PROGRAM PRESS R. ?R  
DO YOU WISH TO (E)XIT THE PROGRAM OR (R)UN IT AGAIN? ?E

## TRUSS ANALYSIS

This program serves as a beginning tutorial for a student in Structural Analysis to assist him in learning the Method of Joints.

### Example:

Learn how to analyze a truss using the Method of Joints. Sample problems are generated by the computer.

Run:

### STRUCTURAL ANALYSIS - TRUSSES METHOD OF JOINTS

- 1) SIGN CONVENTION
- 2) BEGINNING TUTORIAL
- 3) ADVANCED TUTORIAL
- 4) PROBLEM SOLVING - COMPUTER GENERATED
- 5) CONSTRUCTION GAME
- 6) QUIT

??

THE BEGINNING TUTORIAL WILL RUN THROUGH A STEP-BY-STEP STRUCTURAL ANALYSIS OF A TRUSS USING THE METHOD OF JOINTS.

AN IMAGINARY SECTION MAY BE PASSED AROUND A JOINT IN A TRUSS, ISOLATING IT FROM THE REST OF THE TRUSS. THE JOINT WILL BECOME A FREE BODY.

THE EQUATIONS,  $\sum X=0$  AND  $\sum Y=0$ , MAY BE APPLIED TO IT TO DETERMINE THE FORCE IN EACH MEMBER. NO MORE THAN 2 UNKNOWNNS CAN BE DETERMINED AT A JOINT WITH THESE 2 EQUATIONS.

THE MOST IMPORTANT THING FOR YOU TO REMEMBER IS THAT YOU ARE CONCERNED WITH ONLY ONE JOINT AT A TIME. IGNORE ALL OTHERS WHEN WORKING WITH IT.

TO DETERMINE RIGHT AND LEFT REACTIONS USE THE METHOD OF PROPORTIONS WHERE EACH REACTION EQUALS A PORTION OF EACH LOAD.

THE LEFT REACTION = SUM OF ((DISTANCE FROM LOAD TO RIGHT REACTION)/TOTAL LENGTH) \* LOAD

THE RIGHT REACTION = SUM OF ((DISTANCE FROM LOAD TO LEFT REACTION)/TOTAL LENGTH) \* LOAD

$E(Y) = (40 \text{ FT}/80 \text{ FT}) * 20 \text{ K} = 10 \text{ K UP}$   
 $C(Y) = (40 \text{ FT}/80 \text{ FT}) * 20 \text{ K} = 10 \text{ K UP}$

USE METHOD OF JOINTS TO FIND ALL FORCES.

CONSIDER JOINT C:  
 SLOPE OF LINE CA = 20:20 = 1:1

RESOLVE CA INTO X AND Y FORCES:

SUM F(Y) UP = SUM F(Y) DOWN  
 $CA(Y) = -C(Y) = -10 \text{ K DOWN}$

SINCE SLOPE = 1:1,  $CA(X) = CA(Y) = -10 \text{ K LEFT}$   
SINCE SUM F(X)=0,  $CD(X) = -CA(X) = 10 \text{ K RIGHT}$

SINCE THE TRUSS IS SYMMETRICAL, JOINT E WILL BE EXACTLY EQUAL TO JOINT C.  
SHOULD WE ANALYZE JOINT A OR D NEXT? ?A  
JOINT A IS THE CORRECT ANSWER.

WE ALREADY KNOW THE FORCES IN AC.  
THERE IS ONLY ONE UNKNOWN VERTICAL FORCE.  
 $SUM F(Y) = AC(Y) + AD(Y) = 0$   
 $AD(Y) = ? \quad ?-10$   
 $AD(Y) = -10$  IS THE CORRECT ANSWER.

DUE TO THE 1:1 SLOPE,  $AD(X) = AD(Y) = 10 \text{ K RIGHT}$ .  
 $AB(X) = ? \quad ?-20$   
 $AB(X) = -20$  IS THE CORRECT ANSWER.

SINCE THE TRUSS IS SYMMETRICAL, JOINT B WILL BE EXACTLY EQUAL TO JOINT A.

WE WILL FINISH WITH JOINT D.  
SINCE  $DC = -CD$  AND  $DA = -AD$ , JOINT D IS ALREADY SOLVED FOR US.

USE THE PYTHAGOREAN THEOREM TO RESOLVE THE X AND Y FORCES INTO THEIR DIAGONAL COMPONENTS.

A PLAN FOR THE INTEGRATION OF MICROCOMPUTERS  
INTO THE CIVIL ENGINEERING CURRICULUM  
AT KANSAS STATE UNIVERSITY

by

MICHELE C. PERRIN

B.S., Kansas State University, 1982

---

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the  
requirements for the degree

MASTER OF SCIENCE

Department of Civil Engineering

KANSAS STATE UNIVERSITY  
Manhattan, Kansas

1983



## ABSTRACT

This study was undertaken to determine the best method to use in the incorporation of microcomputers into the Civil Engineering curriculum at Kansas State University. The microcomputer can be used to the best advantage when three modes of operation are emphasized simultaneously. First, the microcomputer should be used as a demonstration tool using dynamic, color and sound software to enhance classroom presentations. Second, the microcomputer should be used as a design aid using predeveloped programs to allow students to explore multiple solutions to problems. Third, the microcomputer should be used as a calculating and programming tool to ease the burden of time constraints resulting from iterative calculations. It is recognized that the microcomputer can prove useful as a word processing tool, but due to the shortage of equipment, this has not been listed as a major area of emphasis.

To help attain these goals, recommendations have been presented as to necessary equipment purchases. In addition, an outline description for a course in microcomputer programming and predeveloped software programs have been included in the Appendices.